CONDITION-INFORMED PREDICTION AND RESOURCE-AWARE SELECTION FOR INORGANIC MATERIALS SYNTHESIS PLANNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Materials synthesis planning (MSP) connects AI-driven materials discovery to experimental realization by proposing precursors, synthesis operations, and processing conditions for a target material. Existing approaches predict these components separately or in pairs and treat conditions only as a prediction target. Consequently, MSP remains centered on recipe prediction: it leaves unexplored how conditions can serve as process context for constructing synthesis routes, and it does not address which candidate routes should be executed under limited experimental resources. We propose CondMSP, which extends MSP from recipe prediction to execution-aware planning. CondMSP predicts temperature, time, and atmosphere by combining a joint predictor with variable-specific experts, and then uses these predictions both as process context, grounding the generation and retrieval of operation sequences, and as execution requirements, specifying which thermal stages can share a furnace run and at what cost. This specification is held fixed across all compared methods; within it, CondMSP ranks routes across precursor sets by a common utility and, under furnace budgets, jointly selects routes and allocates their thermal stages by constrained optimization. CondMSP improves condition and operation prediction and, under this shared constraint, raises reference-recipe recovery from 27.9% to 32.9% at three simulated furnace runs per target and from 34.7% to 40.0% at 30 trials, while a new evaluation set of 2,344 targets supports generalization beyond oxide-dominated training chemistry.

1 INTRODUCTION

Materials synthesis planning (MSP) translates AI-driven materials discovery into experimental realization by recommending a synthesis procedure: (1) the precursors, (2) operation sequences, and (3) processing conditions needed to synthesize a target material. For inorganic materials, these choices are interdependent: the same target can be synthesized from different precursor combinations, while thermal and atmospheric conditions influence the phases that eventually form(Rao & Gopalakrishnan, 1997; West, 2022). For example, even with the same precursors, an unsuitable heating temperature can leave a reaction incomplete or favor unwanted phases (Huo et al., 2022). Consequently, choosing one component of the synthesis plan constrains the others, so that none can be determined well in isolation. Moreover, limited experimental trial budgets and equipment availability require choosing which of the many possible routes to test. Owing to this complexity, generalized synthesis rules remain difficult to establish, and route design has long relied on empirical knowledge and expert judgment(Jansen, 2002).

Existing AI-based methods have largely addressed these components individually: precursor prediction (PP) proposes precursor sets for a target (He et al., 2023; Noh et al., 2024; Pan et al., 2026b; Prein et al., 2025b;a; Prein et al.); synthesis operation prediction (SOP) generates operation sequences from the target and its precursors (Na, 2023); and synthesis condition prediction (SCP) predicts conditions such as heating temperature and time (Huo et al., 2022; Karpovich et al., 2023; Prein et al., 2024; Pan et al., 2026a). Predicting components in isolation, however, does not account for how they depend on one another, and recent methods therefore connect several of them: SPENDE predicts operation sequences and their associated engineering conditions for thermoelectric materials (Na, 2023), while MSP-LLM pairs predicted precursor sets with operation sequences to construct candidate synthesis routes (Noh et al., 2026). These approaches improve component prediction, but leave two questions open: (1) they either produce conditions as outputs or do not predict them at all, leaving unexplored

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.

how conditions can serve as process context for SOP, even though conditions and operations are interdependent; and (2) they stop at proposing candidates for a target, leaving unaddressed *how a fixed experimental budget should be allocated across the resulting synthesis routes.*

First, synthesis conditions can provide process context for operation prediction. Operation sequences specify what is done to the precursors, while conditions such as temperature, time, and atmosphere describe the regime in which those operations are carried out. In practice, the two are closely linked; high-temperature treatments, for example, accompany operations such as sintering. Analyses of text-mined recipes have likewise reported systematic relationships between heating procedures and their temperatures (Huo et al., 2022). Making the anticipated processing regime explicit therefore gives operation prediction a physically meaningful intermediate to build on, linking predicted operations to concrete process parameters.

Second, existing MSP stops at proposing candidates and thus remains primarily a recipe-prediction problem: it asks which synthesis routes are plausible, not which of them should be executed under limited experimental resources. For a single target, PP can propose several precursor sets, each with multiple candidate operation sequences, yielding a pool of alternative routes, and allocating a finite budget across them is a planning decision distinct from predicting their components: it requires weighing how strongly each route is supported by the predictions against the resources required for its execution. Batch-processing platforms such as self-driving laboratories make the resource side explicit, since equipment use depends on which experiments are executed together (Szymanski et al., 2023). Which routes can share furnace capacity depends on their synthesis conditions: routes with compatible temperature, time, and atmosphere requirements can be batched together, whereas repeated heating introduces additional thermal stages that each require capacity. The resource cost of a route therefore depends on which other routes are selected with it. With a trial budget alone, selection reduces to ranking routes by a utility that must be comparable across precursor sets; once routes share furnace capacity, it becomes a budget allocation problem that ranking alone does not solve. Formulating it as such for each target extends MSP from predicting recipes to planning their execution.

In this work, we propose CondMSP, an MSP framework that moves from predicting recipes to planning their execution: rather than treating synthesis conditions only as prediction targets, it predicts them and uses them both as process context for constructing synthesis routes and as execution requirements for deciding which routes to attempt under limited experimental resources. CondMSP first predicts temperature, time, and atmosphere from the target–precursor pair by combining a joint condition LLM with variable-specific experts. The predicted conditions then enter SOP through two complementary paths: condition-grounded generation uses them as context for proposing operation sequences, while condition-aware retrieval weights precursor-overlap evidence from recorded syntheses by condition similarity, and their fusion yields a unified SOP ranking. Finally, CondMSP combines PP confidence and SOP rank into a route utility and selects routes by constrained optimization under trial, furnace-run, and furnace-hour budgets, with predicted conditions and thermal stages determining execution costs. Extensive experiments show that CondMSP improves reference-recipe recovery (exact match of precursor set and operation sequence) from 27.9 to 32.9% under three simulated furnace runs per target and from 34.7 to 40.0% at 30 trials, compared with the strongest baseline. These gains have two sources: as process context, predicted conditions raise operation-sequence coverage; route selection spreads trials across precursor sets by a common utility and, under furnace budgets whose compatibility and cost the predicted conditions define for all methods, recovers routes that utility ranking alone misses. On a new dataset of 2,344 targets, this advantage persists without retraining, supporting generalization beyond the oxide-dominated training distribution.

Our contributions are as follows:

- We formulate MSP as an execution-aware planning problem that extends beyond recipe prediction to selecting which synthesis routes to execute under limited experimental resources. Predicted synthesis conditions serve both as process context for route construction and as execution requirements that define the resource constraint under which routes are selected.
- We develop condition-aware SOP that uses predicted conditions to guide operation-sequence generation and to weight evidence from recorded syntheses, fused into a unified ranking, and a resource-aware selector that allocates trials across routes, balancing precursor sets against SOP candidates within each set, under condition-dependent furnace constraints.
- Extensive experiments demonstrate that CondMSP improves synthesis condition and operation prediction over existing methods and achieves higher reference-recipe recovery under limited trial and simulated furnace budgets. We further construct a new evaluation dataset of 2,344 targets to assess generalization under a shifted chemistry distribution.

2 PRELIMINARIES

2.1 MATERIALS SYNTHESIS PLANNING (MSP)

Materials synthesis planning (MSP) aims to identify how a target material can be experimentally synthesized. Given a target material m, a candidate synthesis plan comprises a precursor set p, a synthesis operation sequence $o = [o_1, \ldots, o_L]$, and the corresponding synthesis conditions c. Following prior work (He et al., 2023; Noh et al., 2024; Prein et al., 2025b), a target is specified by its composition alone, without structural information. In this work, a synthesis procedure can be factorized into two parts: a synthesis route $r = (p, o)$, and its process conditions c. An experimental plan specifies which candidate routes to attempt and assigns their heating steps to furnace runs (Section 3.3). A thermal stage is a heating step within a route, indexed in execution order; a furnace run processes a batch of at most C samples with compatible stage requirements, where C denotes furnace capacity. Our framework produces the experimental plan in a single shot, without feedback from experimental outcomes.

2.2 PRECURSOR AND SYNTHESIS OPERATION PREDICTION (PP & SOP)

We build upon MSP-LLM (Noh et al., 2026), a recent LLM-based framework that organizes materials synthesis planning into precursor prediction (PP) and synthesis operation prediction (SOP). PP identifies candidate combinations of starting materials for a target material m, producing a ranked list of precursor sets $\mathcal{P}_m = (p_1, \ldots, p_K)$. The autoregressive PP model generates each precursor set as a token sequence, so distinct decoded sequences can represent the same unordered set. For each precursor candidate p_k, SOP predicts how the precursors should be processed by conditioning on both the target and the precursor set, yielding a ranked list of operation sequences $\mathcal{O}_{m,k} = (o_{k1}, \ldots, o_{kJ})$. Operations are drawn from seven labels adopted from the same framework: *mixing*, *heating*, *sintering*, *shaping*, *drying*, *quenching*, and *annealing*, where sintering and annealing are labeled separately from other heating treatments. Pairing p_k with an operation sequence o_{kj} defines a candidate synthesis route $r_{kj} = (p_k, o_{kj})$. We adopt the framework's PP component and extend SOP with synthesis-condition information.

2.3 SYNTHESIS CONDITION PREDICTION (SCP)

The synthesis conditions associated with a route are denoted by $c = (T, t, a)$, where T denotes the maximum heating temperature, t denotes the maximum heating time, and a denotes the synthesis atmosphere (i.e., air/nonair). Because stage-level annotations are incomplete, conditions are summarized at the record level: these maxima provide a common basis for comparing thermal requirements across procedures with different numbers of heating stages. Conditions are not given when a route is proposed, and for a target–precursor pair (m, p_k) they are only partially annotated in the training data. We therefore predict them as $\hat{c}_k = f_{\text{cond}}(m, p_k)$, which does not require operation annotations and can thus be trained on records unusable for SOP (Appendix C). The predicted conditions then serve as process context for operation prediction and, in resource-aware planning, as the execution requirements that define furnace compatibility and cost for every compared method.

3 PROPOSED METHOD: CONDMSP

CondMSP (Figure 1 (a)) performs synthesis planning in three stages. First, for each precursor candidate p_k of a target material m, it predicts synthesis conditions $\hat{c}_k$ using a hybrid condition model that combines shared and variable-specific predictions (Figure 1 (b); Section 3.1). Second, it uses these predicted conditions to guide both generation and retrieval of synthesis operation sequences, whose candidates are fused into a unified SOP ranking (Figure 1 (c); Section 3.2). Third, CondMSP constructs synthesis routes from precursor–operation pairs and allocates the available experimental budget across these routes, accounting for route utility and shared furnace constraints (Figure 1 (d); Section 3.3). The first two stages are applied independently to each precursor candidate, while the final planning stage jointly selects routes across precursor branches.

3.1 SYNTHESIS CONDITION MODELING

Conditions describe the regime in which operations are carried out, and CondMSP uses them both to ground operation prediction and to characterize execution requirements. Temperature, time, and atmosphere are process variables of the same synthesis procedure, which supports a shared representation for their prediction. At the same time, they differ in character: temperature and time are continuous, but time spans a wide range, so the same absolute error is more consequential at short durations than at long ones, and atmosphere is categorical and imbalanced. We therefore combine a joint predictor that captures their shared context with experts tailored to each variable.

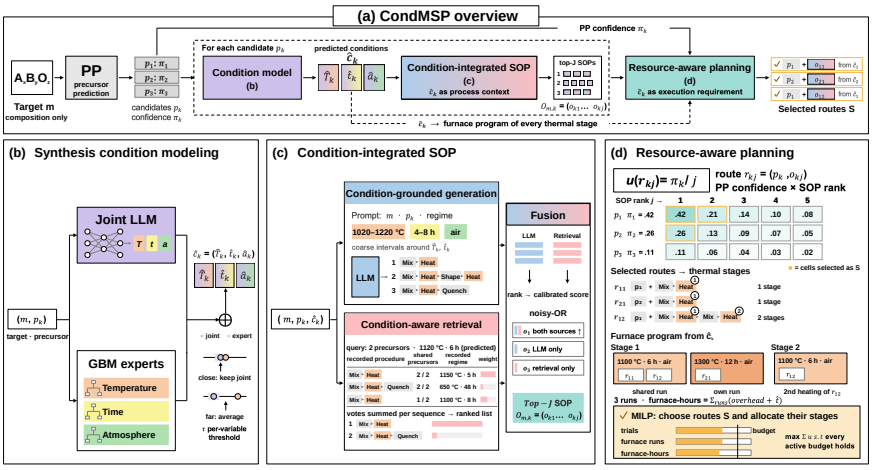

Figure 1: Overview of CondMSP. (a) Predicted conditions connect operation prediction with resource-aware planning. (b) A joint LLM and variable-specific experts predict synthesis conditions. (c) Condition-grounded generation and condition-aware retrieval are fused into an SOP ranking. (d) MILP selects routes and allocates thermal stages under resource budgets. Recipes and numerical values are illustrative.

Joint LLM-based Condition Prediction. We fine-tune a pretrained LLM to jointly predict temperature, time, and atmosphere from target and precursor compositions. This adapts the LLM's pretrained knowledge to synthesis-condition prediction and allows it to learn dependencies among the conditions. Since many synthesis records contain only a subset of these annotations, we train on all observed label combinations, including individual conditions and partially observed combinations, omitting missing fields from the output. This expands the usable training data beyond complete condition triples, enabling the model to learn a richer synthesis context.

Variable-Specific Experts. The joint model's next-token prediction loss does not directly account for numerical error scales or atmosphere-class imbalance. We therefore train a separate gradient boosting machine (GBM)(Friedman, 2001) for each variable using Magpie-based descriptors of the target and precursors(Ward et al., 2016). The temperature expert minimizes absolute error in degrees Celsius; the time expert does so in log space to account for its broad range. The atmosphere expert uses a class-balanced classification loss to account for label imbalance. These objectives provide variable-specific predictions that complement the joint model's shared synthesis context.

Combining Joint and Variable-Specific Predictions. We combine the joint LLM's predictions with those of the variable-specific GBM experts. For each variable, we retain the joint prediction when its absolute difference from the corresponding expert prediction is within a variable-specific threshold τ selected on validation data; otherwise, we average the two. Combination is performed on temperature in °C, time in $\log(1+t)$, and atmosphere in probability space, yielding $\hat{c}_k = (\hat{T}_k, \hat{t}_k, \hat{a}_k)$. Implementation details are provided in Appendix B.3.

3.2 Condition-Informed Synthesis Operation Prediction

The predicted conditions enter SOP through two complementary paths: they ground the LLM's generation and weight retrieved evidence from recorded procedures. We then fuse the candidates from both paths into a unified ranking.

Condition-Grounded LLM Generation. We fine-tune the SOP model on target–precursor inputs augmented with condition descriptions from the frozen condition predictor, which provides the conditions at both training and inference. Conditioning on the processing regime introduces a synthesis-informed inductive bias: a high predicted temperature, for example, cues operations such as sintering. Since condition predictions are imperfect, the prompt describes the processing regime by coarse intervals, whereas retrieval and scheduling use the continuous values. Temperature is represented as $[\hat{T}_k - \Delta_{\text{prompt}}, \hat{T}_k + \Delta_{\text{prompt}}]$, where Δ_{prompt} is a fixed half-width, and time is assigned to a logarithmically spaced interval. Atmosphere is represented as `air` or `nonair` when available (e.g., 1120 °C and 6 h become 1020–1220 °C and 4–8 h with $\Delta_{\text{prompt}} = 100$ °C; Appendix B.4).

Condition-Aware Retrieval. Alongside the LLM's knowledge, we retrieve evidence from recorded synthesis procedures. Shared precursors identify examples of how the same precursors are processed, so we score each candidate sequence by the number of precursors it shares with the query. However,

precursor overlap alone treats procedures with different temperatures or times alike. To distinguish their processing regimes, we weight each procedure's contribution by a soft kernel over the predicted temperature and time that define its thermal regime. Let $\mathcal{D}_{\text{train}}$ denote the database of synthesis procedures in the training set. For query $q = (m, p_k)$ and procedure $d \in \mathcal{D}_{\text{train}}$ with precursor set p_d and operation sequence o_d, we apply the same frozen condition predictor to both, giving a consistent basis for comparison without requiring complete condition annotations in the database. We define the weight

$$w(q,d) = \exp\left(-\frac{|\hat{T}_k - \hat{T}_d|}{b_T}\right) \exp\left(-\frac{|\log(1+\hat{t}_k) - \log(1+\hat{t}_d)|}{b_t}\right), \tag{1}$$

and score each operation sequence by

$$s_{\text{Retrieval}}(o \mid m, p_k) = \sum_{d \in \mathcal{D}_{\text{train}}} \mathbf{1}[o_d = o]\, |p_k \cap p_d|\, w(q,d). \tag{2}$$

Here, $b_T, b_t > 0$ control the temperature and log-time scales, and $|p_k \cap p_d|$ counts shared precursors. Since the predicted conditions are imperfect, the soft kernel gradually reduces the influence of dissimilar regimes while allowing procedures with differing conditions to contribute. The sum retains a frequency prior: sequences repeatedly observed for overlapping precursors and similar predicted conditions receive greater empirical support.

Candidate Fusion. To draw on complementary evidence from the LLM and recorded procedures, we fuse the generated and retrieved candidates into a single ranking. Because scores from the two sources differ in meaning and scale, we map each source's ranks to comparable scores: $q_s(o) = \text{Iso}_s(\text{rank}_s(o))$. Here, Iso_s is a non-increasing isotonic (Zadrozny & Elkan, 2002) fitted separately for each source to validation ranks and binary exact-match labels, ensuring that scores do not increase as rank worsens. The score $q_s(o)$ estimates the exact-match frequency at the candidate's rank in source s. We set $q_s(o) = 0$ for candidates absent from that source. To preserve support from either source while rewarding agreement, we aggregate these scores using noisy-OR:

$$s_{\text{fuse}}(o) = 1 - \prod_{s \in \{\text{Retrieval,LLM}\}} (1 - q_s(o)). \tag{3}$$

The resulting score lies in $[0, 1]$ and requires no additional mixing weight. The top J distinct sequences form $\mathcal{O}_{m,k} = (o_{k1}, \ldots, o_{kJ})$.

3.3 Resource-Aware Materials Synthesis Planning

Generating candidate routes leaves a downstream planning decision: which routes to attempt under limited experimental resources. A trial budget limits the number of routes attempted, while furnace-run and furnace-hour budgets account for equipment use. Predicted conditions specify the execution requirements used to assess sharing between thermal stages from different routes. We formulate this decision as a constrained optimization problem that jointly selects routes and allocates their thermal stages to maximize total utility under resource budgets.

Planning formulation. For target m, let $\mathcal{R}_m$ contain routes $r_{kj} = (p_k, o_{kj})$ from the precursor candidates and their fused SOP lists. We select the route set S that maximizes total utility within the imposed resource budgets:

$$\max_{S \subseteq \mathcal{R}_m} \sum_{r \in S} u(r) \quad \text{s.t.} \quad C_q(S; \Delta_F) \le B_q, \qquad q \in \mathcal{Q}, \tag{4}$$

where S is the route set selected for target m, $u(r)$ is route utility and $C_q(S; \Delta_F)$ is the cost of selected routes under resource q, with budget B_q. Here, $\mathcal{Q} \subseteq \{\text{trial}, \text{run}, \text{hour}\}$ refers to the specific type of imposed resource constraint, and Δ_F is the temperature tolerance for sharing a furnace run.

Route utility. To compare candidate routes across precursor sets, we define route utility as the product of PP confidence and reciprocal rank in the corresponding fused SOP list:

$$u(r_{kj}) = \frac{\pi_k^{\text{PP}}}{j}, \qquad \pi_k^{\text{PP}} = \sum_{z \in \mathcal{Z}_k} \exp\big(\ell(z \mid m)\big), \tag{5}$$

where $\mathcal{Z}_k$ contains the distinct retained PP sequences representing p_k after precursor normalization, and $\ell(z \mid m)$ sums the token log-probabilities of z under the PP model. Summing over $\mathcal{Z}_k$ aggregates confidence across decoded representations of the same precursor set, while $1/j$ applies a common depth discount to each fused SOP list. Our resource-aware selection allocates the available budget

across precursor–SOP alternatives according to route utility, balancing exploration across precursor sets with exploration of SOP candidates within each set. Direct fusion-score weighting yields similar recovery in our comparisons (Appendix E.4); we retain reciprocal rank because it uses the fused ordering without requiring comparable fusion-score gaps across precursor sets. Appendix B.6 details sequence scoring and set-level aggregation.

Shared execution costs. Each selected route uses a separate sample, so $C_{\text{trial}}(S) = |S|$. Each heating, sintering, or annealing operation creates a thermal stage; non-thermal operations keep their order but add no furnace cost. Because stage-level annotations are incomplete, every thermal stage of r_{kj} receives the predicted $(\hat{T}_k, \hat{t}_k, \hat{a}_k)$, charging the predicted maximum time at every stage (a repeated-maximum approximation). Since conditions are predicted per precursor set, they determine which precursor branches can share runs and how long each run occupies the furnace, while each route's operation sequence sets its number of thermal stages. Stages execute in rounds by position; those sharing a run must occupy the same position, have identical predicted times and atmosphere classes, and fit within capacity C, and a common setpoint must lie within Δ_F of every stage's predicted temperature. For fixed S, $C_{\text{run}}(S; \Delta_F)$ is the minimum number of compatible runs; each run adds $\alpha + \hat{t}$ furnace-hours (overhead α, shared time $\hat{t}$), giving $C_{\text{hour}}(S; \Delta_F)$. Appendix B.6 details compatibility, allocation, and stage-program sensitivity; $\Delta_F > 0$ is evaluated in Appendix E.5.

Joint selection and allocation. With a trial budget alone, every route costs one trial, so selecting the highest-utility routes up to B_{trial} is optimal; this setting isolates the cross-branch utility of Eq. (5). With furnace constraints, a route's cost depends on the other selected routes, so a mixed-integer linear program (MILP) jointly selects routes and allocates all their thermal stages, admitting a route only if all its stages fit every imposed budget. Whether these plans recover reference recipes is measured separately. Appendix B.6 gives the MILP, solver limits, and feasible-incumbent handling; Appendix E.5 compares greedy selection on the same pool and cost model.

4 EXPERIMENTS

We evaluate CondMSP along three questions: (1) whether the hybrid condition model improves synthesis-condition prediction (Section 4.1), (2) whether predicted conditions provide useful context for synthesis operation prediction (Section 4.2), and (3) whether resource-aware allocation improves recovery of literature-supported synthesis routes under limited experimental budgets (Section 4.3). CondMSP uses the same frozen PP model as MSP-LLM, isolating the effects of condition-informed operation prediction and route selection (PP accuracy in Appendix D.4).

Datasets. Our main benchmark is curated from the text-mined inorganic synthesis recipes of Kononova et al. (2019), following Noh et al. (2026). It is split into 8,295 training, 932 validation, and 918 test targets with no target shared across splits, and each test target has one reference precursor set and operation sequence. Condition prediction additionally uses 27,110 training records annotated with at least one of temperature, time, or atmosphere; only 11,277 (41.6%) contain all three, which motivates training on all observed label combinations. We also construct a new evaluation dataset from the 80,806 solid-state synthesis records released by Lee et al. (2025). After excluding targets present in the original splits or the condition-training corpus and curating the recorded operations, we obtain 2,344 new targets with 2,482 reference recipes, each pairing a precursor set with an operation sequence; 108 targets have multiple references, which differ in their operation sequences, precursor sets, or both, reflecting the one-to-many relationship between a target and its valid syntheses (Pan et al., 2026a). Relative to the original training recipes, the new references contain far fewer oxide targets (32.6% versus 92.5%) and more all-elemental precursor sets (64.6% versus 6.6%). The pipeline trained on the main benchmark is applied without retraining, testing generalization beyond the chemistry that dominates its training data. More details are provided in Appendix C.

Implementation details. Our trainable LLM components use LLaMA-3.1-8B-Instruct (Grattafiori et al., 2024) with QLoRA (Dettmers et al., 2023; Hu et al., 2021). SOP and planning results report the mean $\pm$ sample SD over three seeds, except for condition-aware retrieval, which is deterministic. More details are provided in Appendix B.

4.1 SYNTHESIS CONDITION PREDICTION (SCP)

Baseline methods. We compare against the predictor of Huo et al. (2022), SyntMTE (Prein et al., 2025b), Karpovich et al. (2023), and our GBM experts without the joint LLM. Variables that a baseline does not predict are marked N/A. Adaptations and training-data variants are detailed in Appendices D and E.1.

Metrics. Given reference precursors, we evaluate maximum temperature and time using mean absolute error (MAE) and root mean squared error (RMSE). Atmosphere prediction uses macro-F1 over air/nonair.

Results. Table 1 shows that combining the joint LLM with variable-specific experts improves prediction over either component alone. Both combination strategies outperform the individual predictors, suggesting that the joint LLM and variable-specific experts provide complementary predictive information.

Table 1: Synthesis condition prediction.

	Temperature		Time		Atmosphere
Model	MAE ↓	RMSE ↓	MAE ↓	RMSE ↓	Macro-F1 ↑
Huo et al. (2022)	151.76	195.29	13.11	25.64	N/A
SyntMTE(Prein et al., 2025b)	171.79 (± 8.71)	217.60 (± 11.65)	N/A	N/A	N/A
Karpovich et al. (2023)	144.36 (± 0.23)	188.85 (± 0.03)	15.35 (± 0.25)	25.31 (± 0.34)	N/A
GBM experts	143.42	190.88	12.93	25.47	54.07
Joint LLM	138.68 (± 3.72)	203.13 (± 1.82)	13.92 (± 0.68)	26.58 (± 1.22)	63.56 (± 4.47)
CondMSP (mean)	131.89 (± 3.18)	**182.01** (± 0.95)	12.46 (± 0.28)	**24.40** (± 0.61)	64.27 (± 0.10)
CondMSP (threshold)	**131.77** (± 3.15)	182.03 (± 0.87)	**12.43** (± 0.35)	24.41 (± 0.69)	**64.58** (± 0.53)

4.2 SYNTHESIS OPERATION PREDICTION (SOP)

Baseline methods. The retrieval-based baseline MPC adapts the material-similarity retriever of He et al. (2023). We also compare with SPENDE (Na, 2023). LLM baselines include MSP-LLM (Noh et al., 2026), fine-tuned GPT-4o mini, and GPT-5.4 mini with zero-shot or 100-shot prompting. MSP-LLM uses the same backbone and SOP training recipe as our generator, without synthesis-condition inputs. CondMSP (w/o conditions) combines MSP-LLM with our precursor-overlap retriever without condition weighting, retaining the same rank-calibration and fusion procedures. Appendix D specifies these adaptations and prompting settings.

Table 2: Synthesis operation prediction.

Method	Top-1	Top-3	Top-5	Top-10	Top-20	Top-30
Specialized Models						
MPC(He et al., 2023)	3.74 (± 0.38)	9.69 (± 0.22)	13.58 (± 0.82)	19.46 (± 0.38)	25.82 (± 0.66)	32.39 (± 0.45)
SPENDE(Na, 2023)	6.68 (± 0.66)	N/A	N/A	N/A	N/A	N/A
MSP-LLM(Noh et al., 2026)	10.71 (± 0.13)	18.92 (± 0.23)	22.88 (± 0.76)	30.28 (± 0.19)	40.23 (± 0.56)	46.51 (± 0.29)
Large Language Models						
GPT-5.4 mini (zero-shot)	0.00 (± 0.00)	0.05 (± 0.08)	0.16 (± 0.23)	2.72 (± 0.15)	7.73 (± 1.08)	11.00 (± 0.46)
GPT-5.4 mini (100-shot)	11.06 (± 0.08)	13.45 (± 0.08)	15.20 (± 0.08)	19.50 (± 0.31)	24.89 (± 0.85)	28.27 (± 0.85)
GPT-4o mini (fine-tuned)	10.97 (± 0.13)	11.55 (± 0.38)	12.93 (± 0.33)	17.79 (± 0.62)	25.53 (± 0.77)	28.32 (± 0.50)
CondMSP (w/o conditions)	11.11 (± 0.00)	19.14 (± 0.17)	24.36 (± 0.41)	32.64 (± 0.44)	42.16 (± 0.50)	47.02 (± 0.67)
CondMSP	**11.40** (± 0.13)	**20.12** (± 0.74)	**24.80** (± 0.38)	**34.24** (± 1.09)	**43.72** (± 0.85)	**49.06** (± 0.49)

Metrics. Top-k exact match is the fraction of targets whose reference operation sequence appears exactly, including order, among the top k candidates. We report $k \in \{1, 3, 5, 10, 20, 30\}$. All methods receive reference precursors, isolating operation prediction from precursor-selection errors. Supplementary NED similarity, LCS-F1, and multiset-F1 scores are defined and reported in Appendix E.2.

Results. Predicted conditions improve SOP coverage: they raise the rate at which the reference sequence appears among the candidates, and the improvement grows with candidate depth (Table 2). CondMSP outperforms MSP-LLM at every reported k, and since the two share the same backbone and SOP training recipe, the gap reflects only what CondMSP adds to generation. This gap is small at Top-1 (11.40% vs. 10.71%) but widens to 3.96 points at Top-10 (34.24% vs. 30.28%) and 2.55 points at Top-30 (49.06% vs. 46.51%). Comparing with CondMSP (w/o conditions) attributes this widening to conditions rather than to retrieval and fusion alone. Precursor-overlap retrieval and fusion account for most of the Top-10 gain (30.28% to 32.64%) but contribute only 0.51 points at Top-30, whereas conditions add 1.60 points at Top-10 and 2.04 points at Top-30 (47.02% to 49.06%), making them the main source of improvement at the depth downstream planning draws on. Since the conditions are themselves predicted from the target and precursors, this gain does not come from new inputs but from how the processing regime is made explicit: the condition model draws on condition-annotated records, including those unusable for SOP training (Section 2), and its predictions give retrieval a continuous similarity over temperature and time where precursor overlap alone treats all procedures sharing precursors alike (paired confidence intervals in Appendix E.7).

4.3 RESOURCE-AWARE MATERIALS SYNTHESIS PLANNING

Baseline methods. We compare MSP-LLM, which explores SOP candidates for its Top-1 predicted precursor set, with fine-tuned GPT-4o mini and zero-shot/100-shot GPT-5.4 mini. Each API-based model is evaluated both end-to-end and as a two-stage PP→SOP pipeline. The end-to-end variant directly generates precursor–SOP pairs; the two-stage variant first predicts one precursor set and then generates SOP candidates for it, following MSP-LLM's Top-1 strategy. All methods use predicted precursors throughout. CondMSP performs utility-based route selection using MILP, while baselines admit whole routes in their original candidate order. To separate candidate construction from route selection, we also cross the two: MSP-LLM candidates (30 beams for each of the shared PP model's Top-10 precursor sets) with utility-based selection, and CondMSP candidates admitted in order within the Top-1 precursor branch (Appendix D).

Metrics. For each target m, a method selects candidate routes S under a per-target budget, counted in trials (one per selected route) or in furnace runs or furnace-hours. Under furnace budgets, the

Table 3: Material synthesis planning under trial and furnace budgets.

	(a) Trial budget (number of syntheses tried)						**(b) Furnace budget** (runs | hours)								
Method	1	3	5	10	20	30	1 run	2 runs	3 runs	5 runs	10 runs	12 h	24 h	48 h	96 h
GPT-5.4 mini (end-to-end, 100-shot)	5.99 (±0.31)	8.71 (±0.46)	9.91 (±0.31)	12.20 (±0.00)	15.14 (±0.31)	16.72 (±0.85)	10.84 (±0.39)	11.66 (±0.15)	13.24 (±0.54)	14.87 (±0.69)	16.50 (±0.69)	5.45 (±0.15)	9.91 (±0.00)	13.51 (±0.31)	15.09 (±0.85)
GPT-5.4 mini (PP→SOP, 100-shot)	7.19 (±0.31)	8.61 (±0.31)	9.97 (±0.54)	12.69 (±0.08)	15.80 (±0.31)	18.08 (±0.31)	12.64 (±0.77)	12.31 (±0.00)	14.81 (±0.00)	17.27 (±0.54)	18.08 (±0.31)	6.37 (±0.39)	11.22 (±0.77)	15.31 (±0.08)	17.27 (±0.39)
GPT-4o mini (end-to-end, fine-tuned)	7.55 (±0.31)	8.46 (±0.67)	10.20 (±0.33)	14.56 (±0.77)	20.26 (±0.98)	21.57 (±1.04)	11.73 (±0.23)	15.94 (±1.06)	19.39 (±1.00)	21.24 (±1.00)	21.53 (±1.09)	6.64 (±0.11)	13.36 (±0.38)	18.23 (±0.62)	20.73 (±1.14)
GPT-4o mini (PP→SOP, fine-tuned)	8.21 (±0.25)	8.79 (±0.25)	9.84 (±0.33)	13.65 (±0.69)	19.72 (±0.58)	21.60 (±0.54)	11.51 (±1.11)	15.98 (±0.45)	19.21 (±0.74)	21.53 (±0.41)	21.60 (±0.54)	6.32 (±0.11)	12.82 (±0.27)	18.05 (±0.35)	20.73 (±0.77)
Candidate construction × route selection															
MSP-LLM	7.81 (±0.13)	13.94 (±0.00)	16.88 (±0.39)	22.55 (±0.47)	30.17 (±0.39)	34.68 (±0.60)	20.04 (±0.11)	21.57 (±0.33)	27.89 (±0.71)	32.75 (±0.25)	34.68 (±0.60)	11.37 (±0.17)	20.04 (±0.50)	27.92 (±0.44)	32.61 (±0.38)
MSP-LLM (w/ CondMSP selection)	7.81 (±0.13)	14.20 (±0.23)	17.97 (±0.19)	23.86 (±0.68)	31.55 (±0.41)	37.22 (±0.25)	20.23 (±0.17)	26.18 (±0.44)	30.72 (±0.29)	36.20 (±0.49)	40.01 (±0.77)	11.44 (±0.19)	21.28 (±0.67)	30.76 (±0.06)	37.25 (±0.11)
CondMSP (w/ MSP-LLM selection)	8.35 (±0.23)	14.52 (±0.38)	17.65 (±0.11)	25.05 (±0.98)	32.72 (±0.60)	37.07 (±0.45)	21.64 (±0.55)	23.06 (±0.06)	30.28 (±0.76)	35.95 (±0.44)	37.07 (±0.45)	12.42 (±0.11)	21.75 (±0.45)	29.05 (±0.38)	34.68 (±0.25)
CondMSP	**8.46** (±0.23)	**14.78** (±0.41)	**18.81** (±0.17)	**26.18** (±1.04)	**34.42** (±0.11)	**40.01** (±0.35)	**21.71** (±0.56)	**27.78** (±0.33)	**32.90** (±0.47)	**38.78** (±0.19)	**42.85** (±0.38)	**12.49** (±0.06)	**23.20** (±0.47)	**31.99** (±0.38)	**39.22** (±0.11)

thermal stages of the selected routes are scheduled together, with predicted conditions determining compatibility and cost; the number of selected routes is then limited only by the furnace budget and the size of each method's candidate pool. All methods are scheduled with the same predicted conditions and simulated cost model (stages grouped by thermal-stage position, capacity $C = 8$, 2 h overhead per run, exact-temperature sharing with $\Delta_F = 0$), so comparisons isolate candidate construction and route selection. Recipe recovery is the percentage of targets for which some route in S exactly matches both the precursor set and the ordered operation sequence of a single reference recipe (any one reference suffices). Further details are provided in Appendices B.5 and B.6.

Results. CondMSP improves recipe recovery from 27.89% to 32.90% at three furnace runs and from 34.68% to 40.01% at 30 trials (Table 3). At 30 trials, the crossed comparisons show gains from both candidate construction (34.68→37.07% under Top-1 admission; 37.22→40.01% under utility-based selection) and selection (34.68→37.22% with MSP-LLM candidates; 37.07→40.01% with CondMSP candidates). As trial-only selection reduces to utility ranking, this selection gain comes from spreading trials across precursor sets: the Top-1 set misses the reference for 236 targets, and 186 of them have it in the top-10 (Appendix D.4). In the matched Top-10 setting, CondMSP (w/o conditions) reaches 37.87%, so conditions in candidate construction add 2.14 points (Appendix E.9). Under furnace budgets, utility and joint allocation both contribute: at 48 h, Top-1 admission of the same candidates recovers 29.05%, utility-greedy 30.79%, and MILP, which also accounts for shared runs, 31.99% (Appendix E.5). Some baselines lose recovery at larger furnace budgets because ordered admission is recomputed per budget (Appendix E.6).

Table 4: Material synthesis planning on the new target set under trial and furnace budgets. Single and Multi denote single-reference ($n = 2{,}236$) and multi-reference ($n = 108$) targets.

	(a) Trial budget Number of syntheses tried												**(b) Furnace budget** Runs										Hours							
	Single-ref.						Multi-ref.						Single-ref.					Multi-ref.					Single-ref.				Multi-ref.			
Method	1	3	5	10	20	30	1	3	5	10	20	30	1	2	3	5	10	1	2	3	5	10	12	24	48	96	12	24	48	96
GPT-5.4 mini (end-to-end, 100-shot)	3.94 (±0.19)	5.64 (±0.19)	6.42 (±0.22)	7.87 (±0.19)	9.73 (±0.28)	10.73 (±0.13)	**15.74** (±2.62)	17.59 (±3.93)	20.37 (±2.62)	25.00 (±2.62)	25.93 (±2.62)	30.09 (±1.96)	6.48 (±0.38)	7.76 (±0.09)	8.99 (±0.06)	10.00 (±0.09)	10.51 (±0.25)	24.07 (±0.00)	26.85 (±1.31)	27.78 (±1.31)	28.24 (±1.96)	29.63 (±1.31)	2.71 (±0.35)	5.50 (±0.25)	8.32 (±0.06)	9.97 (±0.25)	6.48 (±1.31)	12.04 (±2.62)	20.83 (±3.27)	28.24 (±1.96)
GPT-5.4 mini (PP→SOP, 100-shot)	4.18 (±0.03)	5.64 (±0.06)	7.09 (±0.41)	9.68 (±0.35)	12.61 (±0.19)	14.53 (±0.38)	15.28 (±0.65)	18.98 (±1.96)	**27.31** (±1.96)	30.56 (±0.00)	37.04 (±1.31)	40.74 (±1.31)	8.70 (±0.09)	9.48 (±0.06)	12.14 (±0.09)	14.33 (±0.28)	14.53 (±0.38)	25.46 (±1.96)	31.48 (±1.31)	34.72 (±0.65)	40.74 (±1.31)	40.74 (±1.31)	3.18 (±0.13)	7.00 (±0.16)	10.64 (±0.44)	13.37 (±0.57)	9.26 (±0.00)	14.81 (±1.31)	26.85 (±0.00)	36.57 (±0.65)
GPT-4o mini (end-to-end, fine-tuned)	4.01 (±0.07)	4.22 (±0.11)	4.76 (±0.14)	7.53 (±0.07)	13.07 (±0.83)	17.68 (±1.52)	**15.74** (±0.00)	16.05 (±0.53)	17.28 (±0.53)	20.06 (±1.93)	30.25 (±3.85)	37.65 (±3.74)	5.38 (±0.17)	9.21 (±0.27)	14.52 (±0.68)	17.49 (±1.43)	17.68 (±1.52)	19.14 (±0.53)	22.84 (±1.07)	33.95 (±4.66)	37.65 (±3.74)	37.65 (±3.74)	2.50 (±0.04)	6.51 (±0.31)	11.55 (±0.60)	14.88 (±0.74)	5.25 (±1.41)	13.58 (±1.07)	25.00 (±1.85)	33.64 (±3.25)
GPT-4o mini (PP→SOP, fine-tuned)	3.52 (±0.14)	3.83 (±0.17)	4.28 (±0.09)	6.13 (±0.19)	11.05 (±0.39)	16.26 (±0.44)	13.58 (±0.53)	13.89 (±0.93)	14.20 (±1.41)	18.21 (±0.53)	26.85 (±0.93)	37.65 (±3.51)	4.43 (±0.09)	7.80 (±0.19)	13.06 (±0.58)	16.19 (±0.43)	16.26 (±0.44)	16.67 (±0.93)	20.37 (±1.60)	33.64 (±1.41)	37.65 (±3.51)	37.65 (±3.51)	2.47 (±0.07)	5.87 (±0.30)	10.42 (±0.59)	14.04 (±0.50)	5.25 (±1.07)	13.58 (±4.28)	25.00 (±1.85)	34.88 (±3.51)
Candidate construction × route selection																														
MSP-LLM	4.13 (±0.21)	7.48 (±0.32)	9.08 (±0.12)	11.82 (±0.14)	16.74 (±0.75)	20.63 (±0.92)	13.27 (±1.41)	20.99 (±1.07)	24.07 (±0.93)	28.09 (±0.53)	35.19 (±1.85)	44.75 (±1.41)	9.41 (±0.22)	13.36 (±0.36)	16.35 (±0.91)	20.01 (±0.92)	20.63 (±0.92)	25.00 (±0.93)	29.32 (±0.53)	36.42 (±2.83)	43.21 (±1.41)	44.75 (±1.41)	3.94 (±0.15)	9.35 (±0.22)	14.21 (±0.42)	17.96 (±0.74)	7.10 (±1.41)	16.36 (±1.93)	28.09 (±0.53)	38.58 (±1.07)
MSP-LLM (w/ CondMSP selection)	4.17 (±0.21)	**7.90** (±0.37)	9.50 (±0.14)	12.21 (±0.04)	17.47 (±0.81)	21.50 (±0.87)	13.27 (±1.41)	20.99 (±1.07)	24.07 (±0.93)	28.09 (±0.53)	35.19 (±1.85)	44.44 (±1.85)	9.41 (±0.22)	14.15 (±0.47)	18.13 (±0.93)	21.00 (±1.02)	23.39 (±0.91)	25.31 (±0.53)	29.63 (±0.93)	38.27 (±1.41)	43.21 (±1.41)	45.37 (±2.45)	4.16 (±0.19)	9.97 (±0.34)	15.46 (±0.66)	19.57 (±0.58)	8.02 (±0.53)	16.98 (±1.41)	28.70 (±0.00)	39.20 (±3.25)
CondMSP (w/ MSP-LLM selection)	4.20 (±0.37)	7.27 (±0.05)	9.42 (±0.26)	13.86 (±0.12)	20.54 (±0.70)	26.45 (±0.26)	14.51 (±1.93)	**22.22** (±0.93)	25.31 (±0.53)	**30.86** (±0.53)	37.96 (±0.00)	48.77 (±1.41)	10.79 (±0.17)	15.85 (±0.11)	21.44 (±0.34)	25.94 (±0.12)	26.45 (±0.26)	29.94 (±0.53)	33.02 (±1.07)	37.65 (±0.53)	47.84 (±1.41)	48.77 (±1.41)	4.79 (±0.13)	11.75 (±0.14)	18.68 (±0.14)	23.24 (±0.11)	11.11 (±0.00)	20.37 (±0.00)	34.88 (±2.14)	42.59 (±0.93)
CondMSP	**4.25** (±0.37)	7.65 (±0.04)	**9.88** (±0.24)	**14.04** (±0.15)	**21.27** (±0.72)	**27.55** (±0.31)	14.51 (±1.93)	**22.22** (±0.93)	25.31 (±0.53)	**30.86** (±0.53)	**38.27** (±0.53)	**49.07** (±1.60)	**10.88** (±0.13)	**17.11** (±0.36)	**23.46** (±0.23)	**27.24** (±0.09)	**29.59** (±0.27)	**30.25** (±0.53)	**34.26** (±0.93)	**41.36** (±0.53)	**49.69** (±1.41)	**50.00** (±0.93)	**5.11** (±0.14)	**12.28** (±0.10)	**19.74** (±0.07)	**25.61** (±0.23)	**11.42** (±0.53)	**21.60** (±0.53)	**35.80** (±2.14)	**45.99** (±1.41)

Generalization under distribution shift. The new dataset (Lee et al., 2025) departs substantially from the training distribution, with far fewer oxide targets and many more all-elemental precursor sets. Applied without retraining, CondMSP achieves 30-trial recovery of 27.55% on single-reference targets and 49.07% on multi-reference targets, compared with 20.63% and 44.75% for MSP-LLM (Table 4). Much of this improvement comes from CondMSP's candidates: with MSP-LLM selection, they already reach 26.45% and 48.77%, and utility-based selection adds further gains, particularly under furnace budgets. The advantage also holds in both oxide and non-oxide subsets (Appendix E.3), supporting generalization beyond the oxide-dominated training chemistry. Zero-shot results for both datasets are reported in Appendix E.10.

4.4 Ablation studies

Condition prediction. Table 5(a) examines joint prediction and expert combination separately. Adding variable-specific GBM experts improves both separate and joint LLM predictors. Combining the joint LLM with these experts achieves the best reported mean scores, supporting their complementary roles in condition prediction.

Operation prediction. Table 5(b) uses reference precursors on all 918 targets to isolate condition use in generation and retrieval. Relative to MSP-LLM trained and evaluated without conditions, condition-grounded generation improves Top-10/30 exact match from 30.28/46.51% to 32.03/47.89%. Adding condition weighting to our precursor-overlap retriever improves these scores from 31.37/46.19% to 34.20/47.82%. Fusing the two sources gives the highest mean at every k, with the clearest gains at Top-20 and Top-30; the Top-10 mean is nearly tied with retrieval alone.

Table 5: Ablation of CondMSP components.

(a) Synthesis Condition Prediction (SCP)

Model	Joint	GBM	Temperature MAE ↓	Temperature RMSE ↓	Time MAE ↓	Time RMSE ↓	Atmosphere Macro-F1 ↑
(A) Separate LLMs	✗	✗	141.42	207.30	14.01	26.33	60.90
(B) Separate LLMs + GBM	✗	✓	132.03	186.17	12.48	24.72	62.46
(C) GBM experts only	✗	✓	143.42	190.88	12.93	25.47	54.07
(D) Joint LLM	✓	✗	138.68	203.13	13.92	26.58	63.56
(E) CondMSP	✓	✓	**131.77**	**182.03**	**12.43**	**24.41**	**64.58**

(b) Synthesis Operation Prediction (SOP)

Model	Conditions	Generation	Retrieval	Top-10	Top-20	Top-30
(A) MSP-LLM	✗	✓	✗	30.28	40.23	46.51
(B) CondMSP generation	✓	✓	✗	32.03	41.18	47.89
(C) CondMSP retrieval (overlap only)	✗	✗	✓	31.37	40.31	46.19
(D) CondMSP retrieval	✓	✗	✓	34.20	41.94	47.82
(E) CondMSP	✓	✓	✓	**34.24**	**43.72**	**49.06**

(c) Route selection on the same candidate pool ($\Delta_F = 0$)

Method	Utility	Trials 5	Trials 10	Trials 30	Furnace runs 1	Furnace runs 5	Furnace hours 24	Furnace hours 96
(A) Order by PP rank	✗	17.65	25.05	37.07	21.64	35.98	22.19	36.82
(B) Order by SOP rank	✗	12.85	15.80	25.89	19.90	29.23	19.75	31.59
(C) CondMSP	✓	**18.81**	**26.18**	**40.01**	**21.71**	**38.78**	**23.20**	**39.22**

Route selection. Table 5(c) compares selection strategies on the same full CondMSP candidate pool with predicted precursors. Ordering by PP rank exhausts the SOP candidates of each precursor set before moving to the next, whereas ordering by SOP rank takes candidates of the same SOP rank across precursor sets before moving to the next rank. At 30 trials, CondMSP reaches 40.01% recovery, versus 37.07% and 25.89% for these orderings, and it also achieves the highest recovery under every reported furnace budget. The trial-budget gain thus comes not from mere diversification but from weighting precursor sets by PP confidence.

4.5 Further Analyses

Table 6: SOP generation with different condition sources at training and inference, on the subset with recorded conditions.

Training	Inference	Top-10	Top-20	Top-30
None	None	24.02 ± 0.11	33.06 ± 0.75	38.61 ± 0.43
Predicted	Predicted	25.08 ± 0.09	34.94 ± 0.43	40.21 ± 0.61
Predicted	Ground truth	26.13 ± 0.71	35.84 ± 0.43	41.49 ± 0.53
Ground truth	Ground truth	**27.11** ± 0.85	**36.82** ± 0.75	**42.92** ± 0.70

Table 7: Top-30 SOP exact match (%) by reference-sequence training frequency. Parentheses show each group's share of test targets.

	Low (28.1%)	Medium (35.7%)	High (36.2%)
MSP-LLM	1.68 ± 0.59	38.01 ± 0.70	89.76 ± 0.80
CondMSP retrieval	1.55	33.84	97.59
CondMSP generation	1.42 ± 1.12	**42.28** ± 1.07	89.56 ± 0.63
CondMSP	**2.07** ± 0.22	38.92 ± 1.98	95.58 ± 0.46

Effect of condition source on generation. Table 6 varies the source of conditions at training and inference. Because recorded conditions exist for only part of the data, all rows retrain the generator on 6,203 training recipes and evaluate on the 664 test targets with recorded conditions, using reference precursors without retrieval, so absolute scores are not comparable to Table 2. Exact match increases monotonically from no conditions to predicted to ground-truth conditions at every k: predicted conditions raise Top-30 from 38.61% to 40.21%, and ground-truth conditions at inference alone add a further 1.28 points, reaching 42.92% when also used in training. Since both sources share the same interval encoding, the generator is responding to the accuracy of the described processing regime rather than to the presence of a condition block, and predicted conditions already recover about 40% of the oracle gain. Predicted conditions thus act as process context for operation prediction, and better condition prediction translates directly into better operation prediction (inference-time interventions in Appendix E.8).

Complementarity across sequence frequencies. Table 7 reports Top-30 exact match with reference precursors, grouping targets by how often their reference sequence appears among the training recipes (Low: 0–4, Medium: 5–79, High: ≥ 80). Generation and retrieval are strong in different frequency ranges. Condition-grounded generation shows its largest gain on medium-frequency sequences, improving from 38.01% for MSP-LLM to 42.28%, whereas condition-aware retrieval performs best on frequent ones (97.59%). Fusing both, CondMSP exceeds MSP-LLM in every group and is best on rare sequences, although absolute recovery there remains low; it does not fully match the stronger source in the medium- and high-frequency groups, leaving room for improved fusion.

5 Conclusion

This paper presents CondMSP, which extends MSP from predicting recipes to planning their execution by using predicted synthesis conditions both as process context for operation generation and retrieval and as execution requirements for route selection under trial and furnace budgets. CondMSP reaches 49.06% Top-30 exact match on SOP and raises reference-recipe recovery over MSP-LLM from 27.89% to 32.90% at three simulated furnace runs per target and from 34.68% to 40.01% at 30 trials, and its advantage persists without retraining on 2,344 new targets with a shifted chemistry distribution. The resulting per-target planning step provides a basis on which multi-target and closed-loop synthesis planners can build.

ETHICS STATEMENT

This work uses only publicly available, text-mined literature data (Kononova et al., 2019; Lee et al., 2025) and involves no human subjects, personally identifiable information, or animal experiments. We are not aware of conflicts of interest or sponsorship concerns affecting this work.

The plans CondMSP produces are computational hypotheses derived from reported syntheses, not validated procedures. Recipe recovery measures agreement with literature references rather than experimental success, and the furnace costs we report come from a simulated accounting model rather than from laboratory execution (Appendix F). Predicted conditions are point estimates with nonzero error, and inorganic synthesis involves heating, controlled atmospheres, and reactive powders whose safe handling depends on the specific chemistry. Any plan produced by CondMSP should therefore be reviewed by a qualified experimentalist, and checked against the relevant safety and equipment constraints, before execution; CondMSP is intended to prioritize candidate routes for human or automated laboratories under expert supervision, not to replace that judgment.

REPRODUCIBILITY STATEMENT

Source code for CondMSP, including the condition model, the condition-grounded generator and condition-aware retriever, the fusion procedure, and the route-selection solver, is available at `https://anonymous.4open.science/r/CondMSP-FF75`. Section 3 defines the method, and Appendix B gives training, decoding, retrieval, evaluation, and resource-accounting settings, with the planning formulation and solver configuration in Appendix B.6. Appendix C documents both datasets, including the splits of the main benchmark and the construction, curation, and verification of the new evaluation set; Appendix D records how each baseline was adapted and prompted, and Appendix G lists the exact model input and output formats. All reported numbers are means and sample standard deviations over the seeds stated in Appendix B.1, and the uncertainty procedures behind the reported intervals are described in Appendices E.7 and E.9.

AI USE STATEMENT

In this work, we used generative AI tools to implement methods and to clean and reformat a dataset: parts of the implementation and of the reproduced baselines were written with the assistance of an LLM coding assistant, and an LLM was used to curate operation sequences when constructing our new evaluation set from the released records of Lee et al. (2025) (Appendix C.3). We have not used generative AI tools to generate synthetic datasets, to develop theoretical models or conceptual frameworks, to formulate mathematical claims, to provide critical ingredients for proving mathematical claims, to assist in the writing of proofs, to propose or refine the hypotheses of this work, to support qualitative or thematic data analysis, or to interpret the reported results. Assistance with translation is not applicable to this work. We also used generative AI tools to create and edit software code, to create and modify figures, and to edit the manuscript for readability, structure, and layout.

We have reviewed all AI-assisted work. All AI-assisted code was read and tested by the authors, and every number reported in this paper was produced by executing that code on the stated data rather than generated by a language model. The LLM-assisted dataset curation required agreement between two independent responses, followed by a source-fidelity review that rejected invented or reordered steps, and programmatic checks of event provenance and dataset integrity; the protocol and its residual limitations are described in Appendices C.3 and F. AI-assisted figures were checked against the underlying result files. All claims in this paper were formulated by the authors; generative AI tools were used only to revise wording, formatting, and layout of text the authors had already written. We take responsibility for the final content of this work, including text, claims, or artifacts produced with the aid of generative AI.

We note separately that GPT-4o mini and GPT-5.4 mini appear as evaluated baselines (Appendix D), and that one condition-prediction comparison uses training data that the original authors generated with an LLM (Appendix E.1); these are part of the evaluated setting rather than of the authoring process.

REFERENCES

Milad Abolhasani and Eugenia Kumacheva. The rise of self-driving labs in chemical and materials sciences. *Nature Synthesis*, 2(6):483–492, 2023.

Samuel Andrello, Daniel Alabi, and Simon JL Billinge. Representation of inorganic synthesis reactions and prediction: Graphical framework and datasets. *arXiv preprint arXiv:2512.02947*, 2025.

Muratahan Aykol, Joseph H Montoya, and Jens Hummelshøj. Rational solid-state synthesis routes for inorganic materials. *Journal of the American Chemical Society*, 143(24):9244–9259, 2021.

Jaehwan Choi, Seongmin Kim, Junkil Park, Junyoung Choi, Sunggi An, and Yousung Jung. Ai for accelerated materials discovery: From generative design to autonomous realization. *Chemical Reviews*, 126(15):8644–8693, 2026.

Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. Qlora: Efficient finetuning of quantized llms. *Advances in neural information processing systems*, 36:10088–10115, 2023.

Jerome H Friedman. Greedy function approximation: a gradient boosting machine. *Annals of statistics*, pp. 1189–1232, 2001.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

Tanjin He, Haoyan Huo, Christopher J Bartel, Zheren Wang, Kevin Cruse, and Gerbrand Ceder. Precursor recommendation for inorganic synthesis by machine learning materials similarity from scientific literature. *Science advances*, 9(23):eadg8180, 2023.

Riley J Hickman, Malcolm Sim, Sergio Pablo-García, Gary Tom, Ivan Woolhouse, Han Hao, Zeqing Bao, Pauric Bannigan, Christine Allen, Matteo Aldeghi, et al. Atlas: a brain for self-driving laboratories. *Digital Discovery*, 4(4):1006–1029, 2025.

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. *arXiv preprint arXiv:2106.09685*, 2021.

QHJAJ Huangfu and JA Julian Hall. Parallelizing the dual revised simplex method. *Mathematical programming computation*, 10(1):119–142, 2018.

Haoyan Huo, Christopher J Bartel, Tanjin He, Amalie Trewartha, Alexander Dunn, Bin Ouyang, Anubhav Jain, and Gerbrand Ceder. Machine-learning rationalization and prediction of solid-state synthesis conditions. *Chemistry of Materials*, 34(16):7323, 2022.

Martin Jansen. A concept for synthesis planning in solid-state chemistry. *Angewandte Chemie International Edition*, 41(20):3746–3766, 2002.

Christopher Karpovich, Elton Pan, Zach Jensen, and Elsa Olivetti. Interpretable machine learning enabled inorganic reaction classification and synthesis condition prediction. *Chemistry of Materials*, 35(3):1062–1079, 2023.

Edward Kim, Kevin Huang, Adam Saunders, Andrew McCallum, Gerbrand Ceder, and Elsa Olivetti. Materials synthesis insights from scientific literature via text extraction and machine learning. *Chemistry of Materials*, 29(21):9436–9444, 2017.

Seongmin Kim, Yousung Jung, and Joshua Schrier. Large language models for inorganic synthesis predictions. *Journal of the American Chemical Society*, 146(29):19654–19659, 2024a.

Seongmin Kim, Juhwan Noh, Geun Ho Gu, Shuan Chen, and Yousung Jung. Predicting synthesis recipes of inorganic crystal materials using elementwise template formulation. *Chemical Science*, 15(3):1039–1045, 2024b.

Olga Kononova, Haoyan Huo, Tanjin He, Ziqin Rong, Tiago Botari, Wenhao Sun, Vahe Tshitoyan, and Gerbrand Ceder. Text-mined dataset of inorganic materials synthesis recipes. *Scientific data*, 6 (1):203, 2019.

Heeseung Lee, Hyuk Jun Yoo, Hye Su Jang, Byeongho Park, Yang Jeong Park, and Sang Soo Han. Toward self-driving laboratory 2.0 for chemistry and materials discovery. *Materials Horizons*, 13 (10):4712–4739, 2026.

Sanghoon Lee, Kevin Cruse, Viktoriia Baibakova, Gerbrand Ceder, and Anubhav Jain. Text-mined dataset of solid-state syntheses with impurity phases using large language model. *Scientific Data*, 12(1):1969, 2025.

Matthew J McDermott, Shyam S Dwaraknath, and Kristin A Persson. A graph-based network for predicting chemical reaction pathways in solid-state materials synthesis. *Nature communications*, 12(1):3097, 2021.

Gyoung S Na. Artificial intelligence for learning material synthesis processes of thermoelectric materials. *Chemistry of Materials*, 35(19):8272–8280, 2023.

Heewoong Noh, Namkyeong Lee, Gyoung S Na, and Chanyoung Park. Retrieval-retro: retrieval-based inorganic retrosynthesis with expert knowledge. *Advances in Neural Information Processing Systems*, 37:25375–25400, 2024.

Heewoong Noh, Gyoung S Na, Namkyeong Lee, and Chanyoung Park. Msp-llm: A unified large language model framework for complete material synthesis planning. *arXiv preprint arXiv:2602.07543*, 2026.

Elton Pan, Soonhyoung Kwon, Sulin Liu, Mingrou Xie, Alexander J Hoffman, Yifei Duan, Thorben Prein, Killian Sheriff, Yuriy Roman-Leshkov, Manuel Moliner, et al. Diffsyn: a generative diffusion approach to materials synthesis planning. *Nature Computational Science*, 6(4):404–416, 2026a.

Elton Pan, Thorben Prein, Juno Nam, Xiaochen Du, Soojung Yang, Pengfei Cai, Jennifer L.M. Rupp, Rafael Gomez-Bombarelli, and Elsa Olivetti. Synreason: Enhancing synthesis reasoning via reinforcement learning experimental feedback. In *AI for Accelerated Materials Design - ICLR 2026*, 2026b. URL https://openreview.net/forum?id=1kLMpYT9AW.

Thorben Prein, Elton Pan, Alan Albert Piovesana, Behsad Riemer, Elsa Olivetti, Jennifer LM Rupp, et al. Benchmarking augmentation strategies for llm-based solid-state synthesis prediction. In *AI for Accelerated Materials Design-ICLR 2026*.

Thorben Prein, Fuzhan Rahmanian, Kesava Prasad Arul, Jasmin El-Wafi, Menelaos Panagiotis Fotiadis, Jan Heimann, Paul Weinmann, Yifei Duan, Elton Pan, Elsa Olivetti, et al. Reaction graph networks for inorganic synthesis condition prediction of solid state materials. In *AI for Accelerated Materials Design-NeurIPS 2024*, 2024.

Thorben Prein, Elton Pan, Sami Haddouti, Marco Lorenz, Janik Jehkul, Tymoteusz Wilk, Cansu Moran, Menelaos Panagiotis Fotiadis, Artur P Toshev, Elsa Olivetti, et al. Retro-rank-in: a ranking-based approach for inorganic materials synthesis planning. *arXiv preprint arXiv:2502.04289*, 2025a.

Thorben Prein, Elton Pan, Janik Jehkul, Steffen Weinmann, Elsa Olivetti, and Jennifer LM Rupp. Language models enable data-augmented synthesis planning for inorganic materials. *ACS Applied Materials & Interfaces*, 17(51):69221, 2025b.

Chintamani Nagesa Ramachandra Rao and Jagannatha Gopalakrishnan. *New directions in solid state chemistry*. Cambridge University Press, 1997.

Ulrich S Schubert and Nicola Hüsing. *Synthesis of inorganic materials*. John Wiley & Sons, 2019.

Andreas Stein, Steven W Keller, and Thomas E Mallouk. Turning down the heat: design and mechanism in solid-state synthesis. *Science*, 259(5101):1558–1564, 1993.

Nathan J Szymanski, Bernardus Rendy, Yuxing Fei, Rishi E Kumar, Tanjin He, David Milsted, Matthew J McDermott, Max Gallant, Ekin Dogus Cubuk, Amil Merchant, et al. An autonomous laboratory for the accelerated synthesis of inorganic materials. *Nature*, 624(7990):86–91, 2023.

Logan Ward, Ankit Agrawal, Alok Choudhary, and Christopher Wolverton. A general-purpose machine learning framework for predicting properties of inorganic materials. *npj Computational Materials*, 2(1):16028, 2016.

Anthony R West. *Solid state chemistry and its applications*. John Wiley & Sons, 2022.

Bianca Zadrozny and Charles Elkan. Transforming classifier scores into accurate multiclass probability estimates. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 694–699, 2002.

Supplementary Material

- Condition-Informed Prediction and Resource-Aware Selection for Inorganic Materials Synthesis Planning -

A Related Works **16**

B Implementation Details **16**

B.1 LLM training and decoding . . . 16

B.2 Joint condition output and supervision . . . 17

B.3 Composition experts and prediction combination . . . 17

B.4 Condition encoding, retrieval, and fusion . . . 17

B.5 Evaluation . . . 17

B.6 Resource-aware planning implementation . . . 18

C Dataset **20**

C.1 Original benchmark and condition annotations . . . 20

C.2 New target set: source and filtering . . . 20

C.3 Operation curation and verification . . . 21

C.4 Dataset characteristics . . . 21

D Baselines **21**

D.1 Condition prediction . . . 21

D.2 Operation prediction and full planning . . . 22

D.3 API-based LLMs . . . 22

D.4 Precursor prediction (PP) . . . 22

D.5 Adaptation of the interpretable condition predictor . . . 22

E Additional Experiments **23**

E.1 Training data for condition baselines . . . 23

E.2 Soft operation-sequence metrics . . . 24

E.3 New-target results by chemistry . . . 25

E.4 Utility and confidence-dependent allocation . . . 25

E.5 Selection algorithms and sharing tolerance . . . 25

E.6 Budget-wise admission . . . 26

E.7 Paired uncertainty in SOP comparisons . . . 26

E.8 Sensitivity to condition context at inference . . . 26

E.9 Condition context in candidate construction at 30 trials . . . 27

E.10 Zero-shot synthesis planning . . . 27

F Limitations **27**

G Prompt Templates **29**

G.1 Joint condition prediction . . . 29

G.2 Condition-grounded operation prediction 29

A Related Works

Materials Synthesis Prediction. In solid-state synthesis, the phases that form depend on precursor choice, temperature, dwell time, and atmosphere (West, 2022; Rao & Gopalakrishnan, 1997; Schubert & Hüsing, 2019), and route design has long relied on empirical knowledge despite early discussions of rational design and planning of solid-state syntheses (Stein et al., 1993; Jansen, 2002). Thermodynamics-guided approaches rank candidate reactions by computed reaction energies and nucleation barriers (Aykol et al., 2021) or search reaction networks built from computed phase stability (McDermott et al., 2021), but do not output literature-style recipes with operation sequences and processing conditions. Text mining of the synthesis literature has produced large recipe collections (Kim et al., 2017; Kononova et al., 2019; Lee et al., 2025) that enable data-driven prediction of such recipe components. Precursor recommendation has been addressed by materials-similarity retrieval (He et al., 2023), retrieval that combines expert knowledge with attention over retrieved precursor sets (Noh et al., 2024), ranking models (Prein et al., 2025a), elementwise template formulation (Kim et al., 2024b), and LLMs (Kim et al., 2024a), and synthesis-condition prediction by feature-based regressors, reaction graph networks, and data-augmented language models (Huo et al., 2022; Karpovich et al., 2023; Prein et al., 2024; 2025b). Recent work explores reinforcement learning with experimental feedback for precursor reasoning and augmentation strategies for LLM-based synthesis prediction (Pan et al., 2026b; Prein et al.), while DiffSyn models distributions of synthesis compositions and conditions for zeolites (Pan et al., 2026a). For synthesis procedures, graph-based representations of inorganic synthesis reactions have been proposed (Andrello et al., 2025), SPENDE predicts operations and associated conditions for thermoelectric materials (Na, 2023), and MSP-LLM connects precursor prediction with precursor-conditioned operation generation (Noh et al., 2026). CondMSP differs in using predicted conditions as process context for operation generation and as a basis for weighting evidence from recorded procedures, combining both sources into a unified candidate ranking.

Self-Driving Laboratories for Materials Synthesis. Self-driving laboratories integrate predictive models, automated experiments, and feedback to guide synthesis (Abolhasani & Kumacheva, 2023; Szymanski et al., 2023; Hickman et al., 2025), and recent reviews frame autonomous realization as the closing step of AI-driven materials discovery (Lee et al., 2026; Choi et al., 2026). In such platforms, samples are processed in batches, so throughput depends on which experiments can share equipment. CondMSP addresses the per-target step of this pipeline: it formulates the selection of precursor–operation alternatives under trial and furnace budgets, using predicted conditions to determine which thermal stages can share furnace capacity.

B Implementation Details

B.1 LLM training and decoding

The precursor, joint-condition, and operation models use LLaMA-3.1-8B-Instruct with 4-bit quantization and LoRA (Grattafiori et al., 2024; Dettmers et al., 2023). Adapters use rank 16, scaling 32, dropout 0.1, and the query, key, value, and output projections. All models train for three epochs with cosine learning-rate decay, a 0.03 warmup ratio, and loss on assistant response tokens only. The PP model uses learning rate 10^{-4} and effective batch 8; the joint-condition model uses 10^{-4} and batch 4; SOP uses 2×10^{-4} and batch 2. SOP runs use matched training seeds 42, 46, and 51. We first train the condition predictor, then freeze it to generate condition inputs for SOP training and inference.

PP generates 10 beams. The joint-condition model generates five beams, from which the highest-ranked output supplies the deployed condition fields. SOP generates 30 beams with at most 256 new tokens and stops at the completed output list. Sampling is disabled. Raw SOP beams are not deduplicated for standalone evaluation; fusion merges identical operation sequences across its two sources. Inputs contain the target and precursors, with no article title or extracted synthesis-context field. The SOP input also contains precursor-type tags derived from the precursor formulas. Condition and SOP templates appear in Appendix G.

Sensitivity to beam deduplication. Order-preserving deduplication of the saved 30-beam SOP outputs, without replenishing candidates, changes mean exact match across the reported k by at most 0.04 percentage points for MSP-LLM and 0.07 for the CondMSP generator. Top-10 remains 30.28% for MSP-LLM and changes from 32.03% to 32.06% for our generator, indicating negligible sensitivity in these standalone SOP comparisons.

B.2 JOINT CONDITION OUTPUT AND SUPERVISION

The deployed joint-condition model is trained on 27,110 examples and emits available fields in the order `temp`, `time`, `atmos`. Missing training annotations are omitted from the response. Temperature and time supervision use the maximum recorded values. The joint LLM uses six atmosphere labels (air, nitrogen, oxygen, hydrogen, argon, other), mapped to air/nonair downstream.

B.3 COMPOSITION EXPERTS AND PREDICTION COMBINATION

Let ϕ be the Magpie-based composition descriptor and $\bar{\phi}(p) = |p|^{-1} \sum_{v \in p} \phi(v)$ the mean precursor descriptor. Each expert receives

$$h(m, p) = [\phi(m); \bar{\phi}(p); \bar{\phi}(p) - \phi(m)]. \tag{6}$$

The numerical gradient boosting machines use absolute-error loss, 300 iterations, and learning rate 0.06; temperature is fitted in °C and time in $\log(1 + t)$, with t in hours. The atmosphere expert uses 400 iterations, the same learning rate, and class-balanced binary classification. Each expert is fitted on training rows with the corresponding annotation.

For a joint prediction x and the corresponding expert prediction y, the variable-specific threshold rule is

$$G_\tau(x, y) = \begin{cases} x, & |x - y| \le \tau, \\ (x + y)/2, & \text{otherwise.} \end{cases} \tag{7}$$

Temperature is combined directly using Eq. (7). For time, define $\ell_\epsilon(t) = \log(1 + \max(t, \epsilon))$, with $\epsilon = 0.05$ h. The combined prediction is

$$\hat{t} = \exp\big[G_{\tau_t}\big(\ell_\epsilon(t_{\mathrm{LLM}}), \ell_\epsilon(t_{\mathrm{GBM}})\big)\big] - 1. \tag{8}$$

Atmosphere combines the LLM's nonair indicator with the expert's nonair probability, then thresholds the result. Validation selection gives $\tau_T = 20\,^\circ$C, $\tau_t = 0.4$, $\tau_a = 0.3$, and atmosphere threshold $\eta = 0.4$. Numerical gates minimize validation MAE; the atmosphere gate uses macro-F1. If one numerical predictor omits a value, the other supplies it. Atmosphere remains absent from the SOP prompt when the joint LLM omits it.

B.4 CONDITION ENCODING, RETRIEVAL, AND FUSION

The SOP condition block uses a temperature interval of $\hat{T} \pm 100\,^\circ$C and time bins with boundaries $0, 0.5, 1, 2, 4, 8, 16, 32, 64, 128$ h (the last bin is open-ended). Available atmosphere labels are air/nonair. This encoding is fixed during training and inference and across furnace-sharing scenarios. These intervals are prompt settings, not calibrated confidence intervals; retrieval and scheduling use continuous predicted values.

Retrieval uses the 8,295 training procedures, with conditions predicted by the same frozen model used for queries, including procedures with incomplete observed conditions. A precursor-to-procedure lookup computes Eq. (2) by visiting procedures that share at least one precursor. The kernel scales are $b_T = 200\,^\circ$C and $b_t = 1$ in log-time space; atmosphere is omitted from the retrieval kernel. Contributions are summed over training procedures sharing precursors with the query. For each SOP seed, isotonic rank calibration is fitted on 932 validation queries with reference precursors and applied unchanged to deployment queries with predicted precursors. The union contains at most 30 distinct candidates from each source; fusion returns its top 30 sequences. Noisy-OR supplies a monotone ranking score; it is not treated as a calibrated success probability and does not impose source independence.

B.5 EVALUATION

SCP methods are compared on the same evaluation rows for each variable. SOP and planning use all 918 test representatives and report the mean and sample SD across the three SOP seeds, except where stated otherwise. API request variation is distinguished from training-seed variation in Appendix D.

Target-paired confidence intervals resample targets after averaging the matched per-target seed differences. They quantify target sampling uncertainty, separately from the displayed seed SD. Standalone SOP supplies the reference precursor set; full planning supplies predicted precursors to condition prediction, generation, and retrieval. The new target set uses paired references: both the precursor set and operation sequence must match the same reference.

Condition interventions. Table 16 evaluates SOP generation alone on all 918 test targets with reference precursors, using the same frozen condition-trained SOP checkpoints (seeds 42, 46, and 51). Only the inference condition input changes: we retain the original predicted conditions, remove the condition block, or shuffle predicted condition blocks across targets while preserving atmosphere-field presence. The target, precursor input, and checkpoint remain fixed; no model is retrained, and retrieval and fusion are not used. No recorded condition labels are required. We report Top-10 and Top-30 operation-sequence exact match, as the mean and sample SD across the three seeds. Unlike the separately trained configurations in Table 5(b), this comparison tests sensitivity to the condition context at inference.

Condition source in generation. Table 6 varies the condition source at training and at inference. Because recorded conditions are available for only part of the data, all four configurations use the same 6,203 training recipes and the same 664 test targets whose references carry recorded temperature and time; atmosphere is included when recorded and omitted otherwise. The subset is fixed before any model is trained. Each configuration fine-tunes the SOP generator from scratch with the indicated training input (no condition block, predicted conditions from the frozen condition model, or recorded conditions) and is evaluated with reference precursors, without retrieval or fusion, using the indicated inference input. Predicted conditions use the same interval encoding as the main pipeline; recorded conditions are encoded with the same format. We report Top-10, Top-20, and Top-30 exact match as mean and sample SD across three seeds.

B.6 Resource-aware planning implementation

Candidate scope and utility. Planning is performed independently for each target, using all distinct precursor sets from the PP top-10 outputs and up to 30 fused SOP candidates per set. Conditions, generation, and retrieval use each predicted precursor, with no reference-precursor or top-precursor fallback. In Eq. (5), $\mathcal{Z}_k$ groups the distinct retained PP sequences representing the same unordered set after lowercasing and removing spaces. The sequence score sums candidate-token log probabilities without length normalization; set masses are aggregated by log-sum-exp, counting identical serialized outputs once. Ordering uses $\log \pi_k^{\mathrm{PP}} - \log j$, with ties broken by precursor rank and then SOP rank. Normalizing PP masses within a target leaves both the ordering and additive-utility maximizers unchanged.

Predicted thermal programs. Each route is a separate sample; intermediate products are not shared. Every heating, sintering, or annealing operation contributes one of its H_r thermal stages. All stages under precursor candidate p_k use $(\hat{T}_k, \hat{t}_k, \hat{a}_k)$, so a route run alone occupies $H_r \hat{t}_k$ dwell-hours. Atmosphere has two compatibility classes, `air` and `nonair`. Missing atmosphere defaults to `air` for scheduling but remains omitted from SOP prompts; atmosphere is excluded from retrieval weighting. Non-furnace operations retain their sequence order but do not constrain furnace availability in the simulation.

Compatibility and cost. Thermal stages execute in rounds: all first stages finish before any second stages begin, and similarly for later stages. Only stages with the same index h, numerical time t, and atmosphere a can share a run. Within each block $b = (h, t, a)$, a batch A contains at most C samples and requires a common intersection of the intervals $[\hat{T}_r - \Delta_F, \hat{T}_r + \Delta_F]$:

$$\max_{r \in A} \hat{T}_r - \min_{r \in A} \hat{T}_r \leq 2\Delta_F. \tag{9}$$

The executed setpoint is the intersection midpoint, $\theta_A = (\min_{r \in A} \hat{T}_r + \max_{r \in A} \hat{T}_r)/2$. A chain of overlapping neighboring intervals does not ensure a common intersection. Numerical times must agree exactly; SOP prompt intervals and time bins are not used for batching.

Let $n_b^*(S; \Delta_F)$ denote the minimum number of compatible capacity-C batches for selected routes S in block b. Then

$$\begin{aligned} C_{\mathrm{run}}(S; \Delta_F) &= \sum_b n_b^*(S; \Delta_F), \\ C_{\mathrm{hour}}(S; \Delta_F) &= \sum_b (\alpha + t_b) n_b^*(S; \Delta_F). \end{aligned} \tag{10}$$

For a common Δ_F, sorting temperatures and repeatedly taking the largest prefix within capacity and temperature span minimizes the run count. An exchange argument puts the earliest stages in the first

batch without increasing the number of batches and applies recursively. Since each run in block b costs $\alpha + t_b \geq 0$, the same partition minimizes furnace-hours. This rule reconstructs both heuristic and MILP plans, preserving run membership for scoring. A route with no thermal stages consumes one trial and zero furnace resources.

The main simulation uses $\Delta_F = 0$, $C = 8$, and per-run overhead $\alpha = 2$ h. Overhead is a benchmark assumption, not a measured equipment constant. Positive temperature tolerances are evaluated in Appendix E.5. Furnace-hours measure occupancy under these programs, excluding temperature-dependent ramp costs and other laboratory time; parallel furnace count is not constrained.

Invariance to stage programs. Run counts depend on predicted conditions only through which stages at the same position can share a run. At $\Delta_F = 0$, sharing requires identical temperature, time, and atmosphere, so any stage program that sets the stage-h temperature and time to strictly increasing functions $f_h(\hat{T}_k)$ and $g_h(\hat{t}_k)$, such as a lower first-stage calcination temperature or a shorter first-stage dwell, preserves compatibility. At $\Delta_F = 50\,^\circ\mathrm{C}$, the same holds when f_h shifts temperatures by a stage-specific constant, which preserves temperature spans within each stage position. In both cases every run-budget plan and its recovery are unchanged; only furnace-hours can change. Programs that depend on the operation label or on a route's total stage count are not covered by this argument.

Heuristic selectors. Order by PP rank sorts by precursor rank and then SOP rank; Order by SOP rank sorts by SOP rank and then precursor rank. Both use the full CondMSP candidate pool, whereas MSP-LLM selection restricts admission to the Top-1 precursor branch. Utility-greedy sorts by Eq. (5). Each accepts a route only if all its stages fit every active budget after repacking; an infeasible candidate is skipped and the scan continues. A bundle heuristic supplies additional feasible incumbents by considering utility-ordered prefixes with identical program/stage-count profiles and scoring marginal utility against marginal resource use. Overlapping temperature intervals alone do not define a bundle profile.

MILP selection and allocation. Let $x_r \in \{0, 1\}$ select route r. In block b, group stages with the same temperature into classes c, with $\mathcal{R}_c$ the routes contributing a stage to class c. Let $\mathcal{E}_b$ contain the maximal contiguous neighborhoods of classes spanning at most $2\Delta_F$, and let $\mathcal{E} = \bigcup_b \mathcal{E}_b$ retain block identities. For $c \in e$, $z_{ce} \geq 0$ assigns class demand to neighborhood e, $y_e \in \mathbb{Z}_{\geq 0}$ counts its runs, and t_e is their common time:

$$
\begin{aligned}
\max_{x,z,y} \quad & \sum_{r \in \mathcal{R}_m} u(r) x_r \\
\text{s.t.} \quad & \sum_{e : c \in e} z_{ce} = \sum_{r \in \mathcal{R}_c} x_r && \forall c, \\
& \sum_{c \in e} z_{ce} \leq C y_e && \forall e, \\
& \sum_{r} x_r \leq B_{\text{trial}} && \text{if trial} \in \mathcal{Q}, \\
& \sum_{e} y_e \leq B_{\text{run}} && \text{if run} \in \mathcal{Q}, \\
& \sum_{e} (\alpha + t_e) y_e \leq B_{\text{hour}} && \text{if hour} \in \mathcal{Q}, \\
& x_r \in \{0, 1\}, \quad y_e \in \mathbb{Z}_{\geq 0}, \quad z_{ce} \geq 0.
\end{aligned}
\tag{11}
$$

Selecting a route creates demand at every thermal stage, prohibiting partial execution. Every feasible batch lies in a maximal neighborhood, and any group of at most C stages in one neighborhood admits a common setpoint. Continuous z preserves feasibility: for fixed integer x, y, the assignment is a bipartite flow with integer demands and capacities, so a feasible fractional flow has an integral counterpart. Single-neighborhood blocks eliminate assignment variables. Identical resource-demand profiles permit utility-sorted prefix restrictions; these are not applied to profiles that merely overlap in temperature. Selected sets are repacked by Eq. (10), giving all selectors the same cost and executed-setpoint convention.

Solver limits and certification. SciPy/HiGHS(Huangfu & Hall, 2018) uses one CPU thread, a 2 s solver limit per call, relative gap tolerance 10^{-9}, feasibility tolerance 10^{-8}, a fixed solver seed, and presolve disabled. Utilities are rescaled by a common positive factor. We retain the best feasible incumbent from the solver and utility/bundle heuristics, checking all stages, compatibility, capacity, and active budgets. Certification compares the reconstructed incumbent with the solver bound within numerical tolerances; a timeout retains its feasible plan and bound without an optimality claim unless the bound is matched. On the original benchmark, all main repeated-program and matched-pool cases are certified.

Computational cost. Database conditions and the precursor-to-procedure lookup are precomputed. Table 8 reports optimization times for $\Delta_F = 0$, $C = 8$, $\alpha = 2$ h, pooling 918 targets and three seeds without reused compatibility solutions. Timing includes utility/bundle incumbent construction and MILP formulation and solution on one solver thread. These shared-host wall-clock measurements exclude PP, SCP, SOP, and retrieval, and therefore do not measure end-to-end latency.

Table 8: Planning optimization runtime.

Budget	Target–seed cases	Median (s)	95th percentile (s)
3 runs	2,754	0.0109	0.0292
20 runs	2,754	0.0464	0.1337
48 h	2,754	0.0155	0.0597

Reference scoring. References are used only for scoring, never to select routes or change execution programs and costs. Recipe recovery requires a selected route to match the precursor set and ordered operations of the same reference.

C Dataset

C.1 Original benchmark and condition annotations

The original split comprises 8,295 training, 932 validation, and 996 test recipes. Inference uses all test recipes; evaluation uses 918 fixed representatives with one reference recipe per target. The data use seven operation labels: heating, sintering, annealing, mixing, shaping, quenching, and drying. The training split contains 285 distinct precursors. Article-title synthesis context is excluded from every model input.

The 27,110-record condition-training corpus comprises 6,664 records from the SOP training split and 20,446 additional condition-labeled records. The entire condition-training corpus, including these additional records, excludes targets in the original validation and test splits, using the original benchmark's target identifiers. Table 9 reports field availability, with overlapping counts for individual fields and a separate count for complete triples. Training responses omit missing fields rather than requiring complete triples. Temperature and time targets are the maximum recorded values, and atmosphere is evaluated as air/nonair. Missing annotations do not remove targets from SOP or full-planning evaluation.

Table 9: Synthesis-condition annotation availability.

Observed annotations	Records	Share (%)
Temperature	25,963	95.77
Time	24,023	88.61
Atmosphere	13,058	48.17
All three conditions	11,277	41.60

C.2 New target set: source and filtering

We construct a separate evaluation dataset from the 80,806 solid-state synthesis records released by Lee et al. (2025). We remove targets overlapping the original splits or condition-training corpus,

symbolic or unparseable formulas, ambiguous or out-of-vocabulary precursors, elemental targets, records without synthesis actions, and duplicate records. Target identity is the vector of elemental atomic fractions rounded to ten decimals. A conservative element-support filter also excludes potential overlaps hidden in symbolic training formulas. Precursor spellings are normalized to the training vocabulary only when absolute atom counts agree; hydration and stoichiometry are preserved.

C.3 Operation curation and verification

LLM-assisted curation (`gpt-5.4-mini-2026-03-17`, medium reasoning) separates synthesis steps from characterization, repeated descriptions, and actions for other samples. Inputs contain extracted event fragments for the specified target–precursor pair. Each step cites source events, and every event is assigned to a step or an explicit drop reason. Thermal labels follow the recorded action token: sinter maps to sintering, anneal to annealing, and other thermal actions to heating.

Two independent responses must agree on the ordered sequence after collapsing adjacent identical labels. A third source-fidelity review approves, flags, or rejects each record without rewriting it. We reject invented or reordered steps, merges across separately recorded stages, and sequences without a thermal operation, without imposing a preferred recipe length. The resulting dataset contains 2,482 distinct paired references for 2,344 targets: 2,236 single-reference targets and 108 with multiple references.

Paired-reference scoring. Each target contributes one recovery outcome. A selected route must match both the precursor set and the ordered operations of the same reference; a match to any one of the target's paired references satisfies this criterion. Precursor matches and operation matches from different references cannot be combined. For this new-set evaluation, adjacent repeated operation labels are collapsed in both predictions and references. Targets with invalid or empty outputs remain in the denominator. Models and fusion calibrators are frozen before evaluation on the new targets.

Programmatic checks verify target disjointness, precursor-vocabulary membership, unique reference pairs, thermal-operation presence, and event provenance. Recorded condition fragments are retained as evidence but are not used as ground truth in the new-set furnace evaluation.

C.4 Dataset characteristics

Table 10 compares reference-recipe characteristics, using the 918 evaluated representatives for the original test column. The new set contains more non-oxide targets and all-elemental precursor sets, and shorter operation sequences, testing generalization under a chemistry and procedure shift. Here, "oxide" denotes an oxygen-containing target formula; each reference recipe contributes to the profile statistics. We compare the frozen methods on the same new targets under common paired-reference scoring. The split-specific chemistry results are in Appendix E.3.

Table 10: **Recipe characteristics of the original and new data.**

	Original train	Original test (918)	New all refs.	New single refs.	New multi refs.
Recipes	8,295	918	2,482	2,236	246
Oxide (%)	92.5	92.4	32.6	33.9	21.1
All-elemental precursors (%)	6.6	6.3	64.6	63.2	77.6
Mean operation count	4.17	4.31	2.71	2.72	2.66
One-operation recipes (%)	3.1	1.6	23.5	23.4	24.4
Mean precursor count	3.23	3.31	3.56	3.58	3.39
Mean elements per target	4.18	4.25	3.89	3.93	3.61

D Baselines

D.1 Condition prediction

The predictor of Huo et al. (2022) is adapted to temperature and time regression; SyntMTE (Prein et al., 2025b) is adapted to temperature prediction. Both are retrained with their own implementations.

The predictor of Huo et al. (2022) uses the original training split augmented with parseable condition records from the source corpus; SyntMTE uses the original training split. Table 12 additionally reports SyntMTE with LLM-generated training data and a GBM using all available temperature labels. Unsupported tasks are marked N/A. The GBM baseline uses the same composition experts as CondMSP, without the joint LLM.

D.2 Operation prediction and full planning

The MPC baseline adapts the learned material-similarity retriever of He et al. (2023) to return operation sequences from retrieved procedures. SPENDE (Na, 2023) returns one operation sequence, so its Top-k results for $k > 1$ are unavailable. MSP-LLM uses the same backbone and SOP training recipe as our generator but receives no condition block. Standalone SOP uses the reference precursors for every method. In full planning, the original MSP-LLM strategy uses the highest-ranked precursor set from the shared PP model and admits its SOP candidates in order. For MSP-LLM with CondMSP selection in Tables 3 and 4, the same frozen SOP model generates 30 beams for each of the Top-10 predicted precursor sets, using its original prompt and decoding settings without condition context or retrieval fusion. The resulting alternatives receive the utility-based selection of CondMSP. Conversely, CondMSP with MSP-LLM selection admits its fused SOP candidates in order within the Top-1 precursor branch. The Order by PP rank and Order by SOP rank controls in Table 5(c) both use the full CondMSP pool.

D.3 API-based LLMs

GPT-4o mini uses `gpt-4o-mini-2024-07-18`, fine-tuned on the same 8,295 training recipes. Separate fine-tunes predict precursors, operations, or the recipe. SOP and recipe queries draw 30 samples at temperature 0.8, ordered by total sequence log-probability. The two-stage baseline passes the greedy precursor prediction to the SOP fine-tune. Its error bars summarize API request seeds 42, 46, and 51, not independent fine-tuning runs.

GPT-5.4 mini uses zero-shot or 100-shot prompting without fine-tuning; demonstrations are fixed training examples sampled once with seed 0. SOP and recipe queries request 30 ordered candidates in one response at temperature 1.0 with reasoning disabled. End-to-end outputs contain precursors and operations; the two-stage variant first predicts one precursor set at temperature 0. We report request seeds 42 and 46, limiting this baseline to two repetitions because of API cost.

Candidate handling. Duplicate or malformed raw outputs consume slots in trial-budget and standalone SOP evaluations. Before furnace scheduling, malformed routes are excluded and valid duplicates are retained for the baselines. In full planning, all systems use predicted precursors and the full target denominator, including incorrect top-1 precursor predictions. They share the program predictor and furnace cost model at each Δ_F. The original MSP-LLM and API baselines admit whole routes in candidate order. Rows using CondMSP selection use the same utility-based MILP procedure; rows using MSP-LLM selection retain Top-1-branch admission. Main furnace results use $\Delta_F = 0$. Utility-greedy and relaxed-sharing comparisons appear in Appendix E.5, with common fixed-set batch costs at each tolerance. Since system candidate pools differ, the same-pool controls in Table 5 isolate the selection rule.

D.4 Precursor prediction (PP)

Both systems share the frozen MSP-LLM precursor model (training seed 42; Appendix B.1). Table 11 reports precursor-set exact match (%) for its ten-beam outputs on the original test and new targets. Set matching ignores order and repeated formulas, strips whitespace and case, and does not substitute chemically related compounds. Invalid predictions count as misses.

On the original set, top-1 is correct for 682 targets and top-10 contains the reference precursor set for 868. Thus, 186 of the 236 top-1 errors have a correct alternative in the pool, while 50 targets remain uncovered. New-set top-1 and top-10 are 83.19% and 93.90%, respectively.

D.5 Adaptation of the interpretable condition predictor

We adapt the chemistry-only `mp_fraction` XGBoost regressors of Karpovich et al. (2023). Features are target elemental fractions, precursor functional-group fractions, and precursor melting-point

Table 11: **Shared precursor prediction.**

Dataset (targets)	Ranking	Top-1	Top-2	Top-3	Top-5	Top-10
Original test (918)	Original beams	74.29	84.10	88.34	91.83	94.55
	Distinct sets	74.29	84.42	88.78	92.16	94.55
New targets (2,344)	Original beams	83.19	87.41	89.93	91.68	93.90
	Distinct sets	83.19	87.93	90.32	91.98	93.90

Both systems share one frozen PP checkpoint; no between-SOP-seed SD applies. Each target is counted once. The original test uses one reference per target; new targets match any precursor set among their 2,482 reference recipes (2,236 single-reference and 108 multi-reference targets). Original beams retain duplicate slots; distinct sets merge repeats in first-appearance order within the same ten outputs. Invalid candidates count as misses; no additional beams are generated.

minimum, maximum, mean, and range; missing melting points use the released lookup's median. Operation sequences and condition labels are excluded from inputs. Separate regressors predict maximum temperature and time in native units; the implementation has no atmosphere output.

Training combines the original split with raw condition records, as for the predictor of Huo et al. (2022), excluding unparseable formulas and validation/test targets by exact or reduced formula. Following the official aggregation, labels for identical target–precursor sets are averaged, yielding 7,670 temperature and 7,333 time groups. The respective 712 and 717 validation labels are held out. We select configurations by validation RMSE using 100 Hyperopt TPE trials per variable and seed (42, 46) over the official search space:

Parameter	Candidate values
Trees	25, 50, 100, 200
Maximum depth	3, 6, 9
Learning rate	0.1, 0.2, 0.3
Column fraction (tree and level, independently)	0.6, 0.8, 1
Minimum child weight	1, 2, 3
Row fraction	0.7, 0.9, 1

All 996 test inputs receive predictions; evaluation follows Appendix B.5.

E Additional Experiments

E.1 Training data for condition baselines

Table 12 reports the condition-baseline training configurations. All condition-training pools exclude targets in the original validation and test splits, including the 20,446 additional records in our condition corpus. Thus, condition training introduces no validation/test target leakage under the original benchmark's target identifiers. Raw-corpus augmentation gives the adapted baselines access to the same 27,110-record source as the joint model, retaining the required labels and formulas supported by each implementation. Smaller usable sets reflect these annotation and preprocessing requirements. The joint model instead uses available fields without requiring every field to be annotated.

We additionally report SyntMTE trained on the authors' literature and LLM-generated LLMForge recipes, and on their combination with our augmented training data. These are pooled-data adaptations to maximum-temperature prediction: both temperature heads regress the record-level maximum and their outputs are averaged. The combined-data variant uses our validation split; the authors'-data variant retains its source validation split.

The full-label GBM uses all 25,963 temperature-annotated records from our condition corpus, with the same composition descriptors, absolute-error loss, learning rate 0.06, and 300 boosting iterations as the original expert. Unavailable descriptor blocks are represented as missing values, which the histogram-based GBM handles natively, so no labeled record is dropped solely because its formula cannot be featurized. Original formula strings and temperature labels are unchanged. All rows are

evaluated using the common temperature protocol of Table 1. Stars mark the configurations used in that table.

Table 12: Training data for adapted condition baselines.

Temperature				
Model	Training data	Rows	MAE ↓	RMSE ↓
Huo et al. (2022)	Original split	6,228	152.84	197.16
Huo et al. (2022)	+ raw corpus ⋆	11,331	151.76	195.29
SyntMTE	Original split ⋆	6,228	171.79 (± 8.71)	217.60 (± 11.65)
SyntMTE	+ raw corpus	11,331	173.07 (± 2.83)	219.20 (± 2.53)
SyntMTE	Literature + LLM data	30,762	180.85 (± 5.72)	228.02 (± 10.78)
SyntMTE	Above + our corpus	38,390	178.76 (± 2.22)	221.72 (± 0.36)
GBM	Original split	6,228	143.42	190.88
GBM	All temperature labels	25,963	144.48	192.11
CondMSP (threshold)	Joint / expert	27,110 / 6,228	**131.77** (± 3.15)	**182.03** (± 0.87)

Time					
Model	Training data	Rows	MAE ↓	RMSE ↓	±3h ↑
Huo et al. (2022)	Original split	6,255	13.13	25.73	32.38
Huo et al. (2022)	+ raw corpus ⋆	10,627	13.11	25.64	32.06
CondMSP (threshold)	Joint / expert	27,110 / 6,255	**12.43** (± 0.35)	**24.41** (± 0.69)	**40.08** (± 1.46)

All training pools exclude original validation/test targets. Raw-corpus augmentation uses the joint model's 27,110-record source; usable counts depend on required labels and formula preprocessing. LLM data denotes the authors' synthetic LLMForge recipes; "Above + our corpus" combines their literature and synthetic data with our augmented training set. The full-label GBM retains all 25,963 temperature labels, using native missing-feature handling for unavailable descriptors. Stars identify the configurations in Table 1. CondMSP lists joint-model and expert training sizes separately.

E.2 Soft operation-sequence metrics

Table 13 uses the same candidates as Table 2, with reference precursors on all 918 targets. Scores are percentages, with higher values indicating closer agreement. For a prediction o and reference $o^\star$, NED similarity is $1 - d_{\text{edit}}(o, o^\star)/\max(|o|, |o^\star|)$; LCS-F1 is $2\,\text{LCS}(o, o^\star)/(|o| + |o^\star|)$; multiset-F1 replaces LCS with the multiset intersection. Each Top-k score is the best value among the first k candidates. These metrics give partial sequence credit without assessing chemical validity.

Table 13: **Soft operation-sequence scores.**

	NED similarity		LCS-F1		Multiset-F1	
Method	Top-10	Top-30	Top-10	Top-30	Top-10	Top-30
Specialized models						
MPC	69.51 ± 0.10	78.87 ± 0.17	79.38 ± 0.12	86.37 ± 0.15	83.43 ± 0.26	90.10 ± 0.26
SPENDE	N/A	N/A	N/A	N/A	N/A	N/A
MSP-LLM	75.08 ± 0.12	83.67 ± 0.02	83.22 ± 0.02	89.27 ± 0.06	86.81 ± 0.15	92.26 ± 0.07
Large language models						
GPT-5.4 mini (zero-shot)	53.19 ± 0.43	62.86 ± 0.15	65.68 ± 0.38	74.81 ± 0.05	71.18 ± 0.19	81.13 ± 0.14
GPT-5.4 mini (100-shot)	67.22 ± 0.15	75.79 ± 0.34	78.08 ± 0.19	84.58 ± 0.30	82.04 ± 0.03	88.45 ± 0.20
GPT-4o mini (fine-tuned)	62.48 ± 0.15	77.31 ± 0.09	73.71 ± 0.16	85.17 ± 0.16	76.53 ± 0.23	88.83 ± 0.25
CondMSP	76.51 ± 0.29	84.76 ± 0.11	84.35 ± 0.13	90.13 ± 0.09	87.52 ± 0.07	92.97 ± 0.09

SD uses three training seeds for MPC, SPENDE, MSP-LLM, and Ours; three request seeds for GPT-4o mini and two for GPT-5.4 mini. SPENDE returns one sequence: its Top-1 NED, LCS-F1, and multiset-F1 are 50.48 ± 0.75, 63.68 ± 0.60, and 65.87 ± 0.84. Its higher-k scores are unavailable, as in Table 2.

E.3 NEW-TARGET RESULTS BY CHEMISTRY

Table 14 reports paired-reference recipe recovery (%) at 1, 10, and 30 trials on oxygen-containing (oxide, $n = 781$) and non-oxygen-containing targets ($n = 1,563$). The elemental-precursor subset ($n = 1,495$) requires every reference to use only elemental precursors and overlaps these groups. Values are mean ± sample SD across SOP training seeds, or API request seeds for GPT models. At 30 trials, CondMSP exceeds MSP-LLM by 4.44 points on oxide targets and 7.97 points on non-oxide targets, showing that the gain extends beyond the dominant training chemistry.

Table 14: **New-target recipe recovery by chemistry.**

	Oxide			Non-oxide			Elemental precursors		
Method	1	10	30	1	10	30	1	10	30
MSP-LLM	**6.23** (± 0.45)	20.10 (± 0.56)	30.30 (± 0.64)	3.71 (± 0.55)	8.81 (± 0.18)	17.47 (± 1.33)	3.61 (± 0.57)	8.72 (± 0.22)	17.59 (± 1.34)
GPT-4o mini (end-to-end, fine-tuned)	5.93 (± 0.07)	13.83 (± 0.13)	21.25 (± 1.79)	3.86 (± 0.07)	5.25 (± 0.28)	17.27 (± 1.56)	3.97 (± 0.10)	5.35 (± 0.29)	17.81 (± 1.59)
GPT-4o mini (PP→SOP, fine-tuned)	5.68 (± 0.32)	10.84 (± 0.58)	18.57 (± 0.13)	3.13 (± 0.06)	4.61 (± 0.17)	16.59 (± 0.87)	3.14 (± 0.07)	4.59 (± 0.17)	16.88 (± 0.91)
GPT-5.4 mini (end-to-end, zero-shot)	0.06 (± 0.09)	1.28 (± 0.18)	4.74 (± 0.18)	0.58 (± 0.18)	2.21 (± 0.14)	4.29 (± 0.09)	0.60 (± 0.19)	2.27 (± 0.09)	4.41 (± 0.09)
GPT-5.4 mini (end-to-end, 100-shot)	5.19 (± 0.45)	13.25 (± 0.45)	18.44 (± 0.36)	4.13 (± 0.23)	6.37 (± 0.14)	8.22 (± 0.14)	4.21 (± 0.28)	6.09 (± 0.09)	7.83 (± 0.28)
GPT-5.4 mini (PP→SOP, zero-shot)	0.00 (± 0.00)	2.11 (± 0.63)	8.64 (± 0.09)	0.16 (± 0.05)	3.71 (± 0.18)	9.47 (± 0.54)	0.17 (± 0.05)	3.85 (± 0.24)	9.57 (± 0.57)
GPT-5.4 mini (PP→SOP, 100-shot)	4.93 (± 0.09)	11.97 (± 0.09)	17.35 (± 0.09)	**4.57** (± 0.05)	9.98 (± 0.45)	14.94 (± 0.59)	**4.58** (± 0.14)	10.00 (± 0.33)	15.18 (± 0.47)
CondMSP	6.10 (± 0.20)	**21.25** (± 0.26)	**34.74** (± 0.52)	4.03 (± 0.61)	**11.60** (± 0.10)	**25.44** (± 0.33)	3.97 (± 0.61)	**11.37** (± 0.07)	**25.64** (± 0.38)

E.4 UTILITY AND CONFIDENCE-DEPENDENT ALLOCATION

PP mass compares precursor support across branches, while reciprocal SOP rank applies the same gradual depth discount within each branch. The rank already reflects fused generation and retrieval evidence, without requiring fusion-score gaps to represent comparable gains across precursor branches.

Direct fusion-score utility. Replacing π_k^{PP}/j with $\pi_k^{\mathrm{PP}} s_{\mathrm{fuse}}(o_{kj})$ while fixing the candidate pool, MILP, and costs gives similar recovery on the 918-target test set: 40.01% versus 39.87% at 30 trials and 31.99% versus 32.24% at 48 h (reciprocal rank versus direct scores; means over three SOP seeds). All compared plans are certified for their utilities. Reciprocal rank provides a simple common depth discount without requiring comparable fusion-score gaps across precursor sets.

E.5 SELECTION ALGORITHMS AND SHARING TOLERANCE

Utility-greedy and MILP selection. We fix the CondMSP candidate pool, condition predictions, PP masses, SOP ranks, capacity $C = 8$, and overhead $\alpha = 2$ h. Utility-greedy admits whole routes in utility order whenever their stages fit the budget; MILP optimizes the same utility under shared execution costs. MILP was selected by validation recovery, weighting the run- and hour-budget families equally at each tolerance. At $\Delta_F = 0$, greedy and MILP recover 30.25% and 32.90% at three runs, and 30.79% and 31.99% at 48 h, respectively, averaged across three SOP seeds on 918 targets. At one run, greedy is slightly higher (21.86% versus 21.71%). Under trial budgets alone, both use the same utility ranking and recover 40.01% at 30 trials. Both selectors thus optimize the same route utility and coincide under trial budgets; under furnace budgets, MILP generally adds recovery by accounting for shared runs. Tables 3 and 4 cross candidate construction with selection, while Table 5(c) compares selection strategies on the same full candidate pool.

Relaxed temperature sharing. Keeping the same pool, utilities, and exact time, atmosphere, and stage-position matching, $\Delta_F = 50\,^{\circ}\mathrm{C}$ gives greedy/MILP recovery of 31.19/33.66% at three runs and 31.45/32.46% at 48 h, compared with 30.25/32.90% and 30.79/31.99% at $\Delta_F = 0$; trial-only selection is unchanged. Two observations follow. First, the MILP advantage over greedy persists at the looser tolerance (2.47 points at three runs; 1.02 points at 48 h, 95% CI $[-0.18, 2.18]$, where

the interval includes zero), indicating that the gain from joint selection does not depend on exact-temperature matching. Second, relaxing the tolerance raises recovery for both selectors by 0.5–0.9 points, because more stages become batchable; the effect is modest because conditions are shared within each precursor branch and sharing across branches also requires identical predicted times. A larger tolerance does not guarantee higher recovery: the shared setpoint deviates from each route's predicted temperature by up to Δ_F, and the selected routes can change, so a plan that is cheaper under the cost model need not contain more reference routes. We therefore report $\Delta_F = 0$ as the main setting and treat Δ_F as a scenario parameter whose appropriate value depends on how much setpoint deviation the target chemistry tolerates. Confidence intervals use 2,000 target-paired bootstrap draws after averaging the three seed outcomes within each target; they are pointwise.

E.6 Budget-wise admission

Each budget is evaluated independently. Ordered-admission baselines scan their candidate lists from the beginning, admitting a whole route when all its stages fit the current budget. MILP solves the corresponding utility-allocation problem at each budget. We score the returned plan for that budget without substituting a plan evaluated at another budget.

E.7 Paired uncertainty in SOP comparisons

Table 15 reports paired differences for the SOP comparisons in Tables 2. All methods are evaluated on the same 918 test targets with reference precursors. For each target, we first average the exact-match difference over the three matched SOP seeds, then resample the 918 targets with replacement 20,000 times. We report percentile 95% confidence intervals. These pointwise intervals quantify target sampling uncertainty conditional on the fitted models; training variation is reported separately as seed SD in the main tables. No multiple-comparison adjustment is applied.

Relative to CondMSP without conditions, the gains at Top-10, Top-20, and Top-30 are 1.60, 1.56, and 2.03 percentage points with intervals above zero, whereas the intervals at Top-1 to Top-5 include zero. The gains over MSP-LLM are positive at every cutoff. Condition use thus improves exact match mainly at deeper cutoffs, consistent with the coverage effect reported in Section 4.2.

Table 15: Paired differences in SOP exact match (percentage points) on the 918 test targets with reference precursors. Each cell reports the mean difference and its 95% bootstrap confidence interval; positive values favor CondMSP.

Comparison	Top-1	Top-3	Top-5	Top-10	Top-20	Top-30
CondMSP − CondMSP (w/o conditions)	+0.29 [−0.33, +0.91]	+0.98 [−0.04, +2.00]	+0.44 [−0.51, +1.42]	+1.60 [+0.29, +2.94]	+1.56 [+0.40, +2.76]	+2.03 [+0.65, +3.41]
CondMSP − MSP-LLM	+0.69 [+0.00, +1.42]	+1.20 [+0.07, +2.36]	+1.92 [+0.65, +3.27]	+3.96 [+2.29, +5.63]	+3.49 [+1.96, +5.05]	+2.54 [+0.98, +4.14]

E.8 Sensitivity to condition context at inference

Table 16 evaluates SOP generation alone with reference precursors on all 918 test targets, keeping the condition-trained checkpoints fixed and changing only the condition input at inference (protocol in Appendix B.5). Removing the condition block or shuffling predicted conditions across targets lowers exact match at both cutoffs; at Top-30, performance drops from 47.89% to 45.28% and 44.73%, respectively. Shuffled conditions fall below the removed-block setting, indicating that the generator relies on the content of the conditions and their alignment with the target–precursor pair, not merely on their presence. This complements Table 6, which varies the condition source at training and inference on the subset with recorded conditions.

Table 16: Sensitivity of SOP generation to condition context at inference (exact match, %; mean ± sample SD over three seeds).

Inference input	Top-10	Top-30
Conditions removed	29.96 ± 0.33	45.28 ± 0.45
Conditions shuffled across targets	30.28 ± 0.95	44.73 ± 1.09
Target-matched conditions	$\mathbf{32.03} \pm 0.66$	$\mathbf{47.89} \pm 0.38$

E.9 CONDITION CONTEXT IN CANDIDATE CONSTRUCTION AT 30 TRIALS

We compare CondMSP with its variant without condition context in candidate construction, using the same predicted precursor alternatives, precursor masses, and utility-based selection procedure. CondMSP (w/o conditions) combines an SOP generator trained without condition inputs with our precursor-overlap retriever without condition weighting. Both retain the same rank-calibration procedure and fusion rule. Conditions are removed only from SOP generation and retrieval; condition predictions used for resource accounting remain unchanged.

Table 17: Candidate-construction control at 30 trials on the same 918 test targets. Recovery is reported as mean ± sample SD across three SOP seeds. The final row reports the paired difference in percentage points and its 95% confidence interval.

Method	30-trial recovery (%)
CondMSP (w/o conditions)	37.87 ± 0.82
CondMSP	**40.01** ± 0.35
Difference (pp) [95% CI]	+2.14 [0.91, 3.41]

At 30 trials, condition context improves recovery by 2.14 percentage points (Table 17). We average matched differences across the three SOP seeds within each target and resample the 918 targets with replacement 20,000 times. The percentile 95% interval is pointwise, without multiple-comparison adjustment, and quantifies target sampling uncertainty conditional on the fitted models. Differences are computed before rounding.

E.10 ZERO-SHOT SYNTHESIS PLANNING

Table 18 reports the GPT-5.4 mini zero-shot baselines under the evaluation and resource-accounting protocols of Tables 3 and 4. Values are recipe recovery (%, mean ± sample SD over request seeds 42 and 46). Zero-shot recovery stays below 10% on the original test set and on single-reference new targets at every budget, and reaches 28.24% only on the 108 multi-reference new targets at 30 trials.

Table 18: Zero-shot synthesis planning with GPT-5.4 mini under trial and furnace budgets. New (single) and New (multi) denote single-reference ($n = 2{,}236$) and multi-reference ($n = 108$) targets of the new set.

		Trials			Runs			Hours		
Test set	Variant	1	10	30	1	3	10	12	48	96
Original	end-to-end	0.00 (± 0.00)	0.93 (± 0.08)	3.59 (± 0.31)	3.27 (± 0.15)	1.53 (± 0.15)	3.49 (± 0.31)	1.58 (± 0.08)	2.72 (± 0.00)	3.05 (± 0.15)
Original	PP→SOP	0.00 (± 0.00)	2.23 (± 0.08)	7.30 (± 0.31)	6.54 (± 0.15)	1.53 (± 0.31)	7.24 (± 0.23)	2.89 (± 0.39)	4.58 (± 0.15)	6.10 (± 0.15)
New (single)	end-to-end	0.36 (± 0.13)	1.63 (± 0.16)	3.94 (± 0.06)	2.42 (± 0.00)	1.97 (± 0.06)	3.49 (± 0.13)	1.01 (± 0.03)	2.57 (± 0.03)	2.91 (± 0.06)
New (single)	PP→SOP	0.11 (± 0.03)	2.59 (± 0.25)	8.27 (± 0.13)	5.75 (± 0.09)	2.21 (± 0.28)	8.25 (± 0.09)	1.59 (± 0.16)	4.85 (± 0.22)	6.28 (± 0.47)
New (multi)	end-to-end	1.39 (± 0.65)	7.41 (± 0.00)	14.81 (± 1.31)	9.72 (± 1.96)	10.65 (± 1.96)	12.50 (± 0.65)	4.63 (± 0.00)	9.26 (± 1.31)	10.19 (± 0.00)
New (multi)	PP→SOP	0.00 (± 0.00)	15.28 (± 1.96)	28.24 (± 5.89)	18.52 (± 1.31)	13.43 (± 3.27)	27.78 (± 5.24)	6.94 (± 0.65)	17.13 (± 1.96)	23.61 (± 4.58)

F LIMITATIONS

Evaluation scope. The available literature datasets do not provide experimental outcomes for every proposed route. Recipe recovery therefore offers a reproducible retrospective measure of whether a planner selects a recorded precursor–operation pair within a specified budget. This directly tests the prioritization of literature-supported routes, while leaving room for valid alternatives absent from the references. A match does not establish phase purity or synthesis success. The new target set broadens the chemistry and reference coverage of this evaluation, although its references are LLM-curated extractions without expert adjudication. Target disjointness tests transfer to targets

outside the task-specific training data; it does not establish a temporal holdout or absence from foundation-model pretraining.

Resource-model scope. Public large-scale solid-state synthesis datasets (Kononova et al., 2019; Lee et al., 2025) encode materials, operations, and selected reaction conditions, but do not provide the standardized process and equipment metadata required for detailed laboratory scheduling. In particular, ramp profiles, sample loads, crucible- and gas-specific compatibility, and equipment overheads are not jointly annotated in their released schemas. This data gap constrains the granularity at which resource-aware synthesis planning can currently be benchmarked. We therefore construct a common resource model from predicted operation sequences and record-level predictions of temperature, duration, and atmosphere. The formulation makes condition-dependent resource use an explicit part of route selection and enables reproducible evaluation of budget allocation when costs depend on shared furnace capacity. All compared methods use the same compatibility and cost rules, enabling controlled comparisons within these resource settings. Repeating predicted maxima across stages, restricting sharing to the same stage position, and using air/nonair classes simplify laboratory execution. Capacity, overhead, and temperature tolerance are scenario assumptions; predicted costs and compatibility are not guarantees of laboratory feasibility. Resource settings and temperature-sharing sensitivity are specified in Appendices B.6 and E.5. The formulation makes these assumptions explicit and provides a basis for incorporating measured programs and equipment-specific constraints when such data become available.

Candidate coverage and selection. Retrieval is limited to recorded sequences, and selection operates on the upstream candidate pool. This finite-pool setting represents the decision faced after candidate generation: how to allocate a limited budget among the available alternatives. Candidate–selection cross-comparisons and same-pool controls distinguish gains in candidate construction from gains in allocation. Route utility expresses relative priority, so a certified utility optimum need not maximize recipe recovery or synthesis success. Planning is performed separately for each target and does not adapt to experimental outcomes. Batch-processing platforms such as self-driving laboratories (Szymanski et al., 2023) schedule experiments across many targets and revise plans as results arrive; extending our formulation to such settings requires target-level priorities for cross-target batches and outcome-dependent updates for sequential campaigns. Both extensions build on the same ingredients the current formulation already makes explicit: predicted conditions as execution requirements, thermal stages as the unit of furnace sharing, and route utility as the quantity to allocate a budget against. The per-target allocation step established here is therefore the component that a multi-target or closed-loop planner would call at each round, rather than a separate problem.

G PROMPT TEMPLATES

G.1 JOINT CONDITION PREDICTION

Both models use the following system message. Braced fields in the user prompts are filled for each target–precursor pair.

System
You are an expert solid-state chemist planning a material synthesis.

01 Joint condition prediction

User

```
[TASK: conditions] Given the target compound ['{target_formula}'] and
the selected precursors {precursor_list}, predict the solid-state
synthesis conditions: the maximum heating temperature (degrees
Celsius), the maximum heating time (hours), and the heating
atmosphere (one of ['air', 'nitrogen', 'oxygen', 'hydrogen', 'argon',
'other']).
Answer ONLY with a Python list of 'key=value' items in this order
(include only the conditions you can predict): ['temp=<int>',
'time=<number>', 'atmos=<class>']. Example: ['temp=1200', 'time=12',
'atmos=air'].
```

G.2 CONDITION-GROUNDED OPERATION PREDICTION

Temperature and time bounds are generated as described in Appendix B.4. The atmosphere clause is omitted when the joint predictor does not emit atmosphere. MSP-LLM uses the same template without the entire condition block.

02 Condition-grounded SOP

User

```
[CONDITIONS] temp in {T_low}-{T_high} C, time in {t_low}-{t_high} h,
atmos={air_or_nonair} [/CONDITIONS]
[PRECURSOR_TYPES]
types_present: {precursor_types}
[/PRECURSOR_TYPES]
Task:
Given the target compound ['{target_formula}'], first classify its
material group,
then based on the selected precursors {precursor_list},
propose one plausible synthesis operation sequence using only the
following operations:
['heating', 'sintering', 'annealing', 'mixing', 'shaping',
'quenching', 'drying'].
Typically, two or more precursors are required.
The possible material groups are:
['oxide', 'alloy', 'phosphate', 'pyrophosphate', 'halide',
'oxyhalide',
'carbide', 'nitride', 'selenide', 'sulfide', 'other'].
Output format:
[['material group'], ['precursor_type1', 'precursor_type2', ...,
'precursor_typeK'], ['precursor1', 'precursor2', ..., 'precursorN'],
['op1', 'op2', ..., 'opM']]
Return only the Python list of single-quoted strings with no extra
text or explanation.
```