PROMPT OUTLINES AND EXAMPLES FOR EXEOS (COMPLEMENTS SECTION 4 – OUR APPROACH)

NEGIN AYOUGHI, DAVID DEWAR, SHIVA NEJATI AND MEHRDAD SABETZADEH

(a) Prompt Outline for Extracting Problem Objectives from NL Problem Description

> Identify and extract the objective(s) of the optimization problem description. Clearly state what is being optimized. Return the objective as a string.
> Example: {Few-shot examples}
> Natural-Language description of the optimization problem: {Problem description}

(b) Examples of Extracted Problem Objectives

> Maximize total revenue from product sales based on the production quantity for each product type.
> Minimize the inventory cost of unused resources.

Fig. 1. Extracting objectives from problem description in Step 1 of EXEOS.

(a) Prompt Outline for Extracting Problem Constraints from Problem Description

> Identify all constraints described in the problem. Extract each constraint as a separate item in a list of strings. Constraints can be explicit or implied.
> Example: {Few-shot examples}
> Natural-Language description of the optimization problem: {Problem description}

(b) Examples of Extracted Problem Constraints

> Inventory: For each resource, usage cannot exceed available inventory and purchases.
> Budget limit: The total purchase cost must not exceed the purchasing budget.

Fig. 2. Extracting constraints from problem description in Step 1 of EXEOS.

(a) Prompt Outline for Generating Metadata

Identify all parameters mentioned in the optimization problem description. For each parameter, extract the following:
- symbol: a short symbolic name for the parameter.
- definition: a brief explanation of what the parameter represents.
- dimension: a list showing the parameter's dimensionality.

Example: {Few-shot examples}
Natural-Language description of the optimization problem: {Problem description}

(b) Examples of Generated Metadata

symbol: NumProducts
definition: Number of products
dimension: []

symbol: NumResources
definition: Number of resources
dimension: []

symbol: price
definition: Selling price per product
dimension: [NumProducts]

symbol: unit
definition: Units of resources r required per unit of product p
dimension: [NumResources, NumProducts]

symbol: inventory
definition: Initial inventory level for each resource
dimension: [NumResources]

symbol: hold
definition: Inventory cost per unused unit of each resource
dimension: [NumResources]

symbol: buyCost
definition: Purchase cost per unit for each resource
dimension: [NumResources]

symbol: budget
definition: Total purchasing budget
dimension: []

Fig. 3. Metadata generation in Step 1 of EXEOS.

(a) Prompt Outline for Rewriting Problem Description with Parameter References

Rewrite the optimization problem description using symbolic references for all parameters. Use \param{symbol} or \param{symbol}_i style to reference parameters symbolically in the rewritten text.
Example: {Few-shot examples}
Natural-Language description of the optimization problem: {Problem description}

(b) Examples of Problem Description with Parameter References

We have \param{NumProducts} products built from \param{NumResources} resources. The units of resource r needed per unit of product p are \param{unit}. Initial on-hand units for each resource are \param{inventory}. Additional units may be purchased at unit cost \param{buyCost}, with total spend limited by \param{budget}. Each product has selling price \param{price}, and unused resources incur inventory cost \param{hold} per unit. The objective is to choose the production quantity for each product and the purchase quantity for each resource to maximize total revenue and minimize the inventory cost of leftover resources under inventory constraints using \param{unit}, initial inventories \param{inventory}, and limit total purchase cost with \param{budget}.

Fig. 4. Rewritten problem description with parameter references in Step 1 of EXEOS.

(a) Prompt Outline for Extracting Decision Variables from Problem Description

Identify all decision variables mentioned in the optimization problem. For each variable, extract the following:
- symbol: a symbolic name for the variable.
- definition: a brief explanation of what the variable represents.
- dimension: a list showing the variable's dimensionality.
Example: {Few-shot example}
Natural-Language description of the optimization problem: {Problem description}

(b) Examples of Extracted Decision Variables

symbol: x
definition: Production quantity per product.
dimension: [NumProducts]

symbol: y
definition: Purchase quantity per resource.
dimension: [NumResources]

symbol: leftover
definition: Unused inventory per resource.
dimension: [NumResources]

Fig. 5. Extracting decision variables from problem description in Step 1 of EXEOS.

I. Generating Model Instruction: Generate a complete and executable AMPL code that accurately formulates the given structured optimization problem, following the provided guidelines.
II. {AMPL Syntax Rules}
III.{Few-shot examples}
IV. {Structured optimization problem}
V. The previous AMPL code for the following optimization problem failed to run. Using the solver feedback, locate the lines in the model most likely causing the errors and explain each error message. Regenerate corrected AMPL code that fixes all issues and follows the guidelines.
VI. {Previously Generated Optimization Model}
VII. {Solver Feedback}

Fig. 6. Prompt outline for generating an optimization model in Step 3 of EXEOS: When Step 3 is invoked immediately after Step 1 for initial model creation, it uses items **I**, **II**, **III** and **IV** as its prompt. When Step 3 is invoked after Step 4 for model refinement, all items are used in the prompt.

Table 1 lists AMPL syntax for formulating optimization models, derived from the AMPL book [1, 2].

Table 1. AMP Syntax Rules and Guidelines

Category	**Rule**
Set and Parameter Declaration	Declare all sets and parameters before use.
Index Domains	Specify index domains for parameters, variables, and constraints using defined sets.
Unique Names	Assign unique names to each constraint and objective to prevent conflicts.
Expression Parentheses	Enclose complex expressions in summations or constraints in parentheses.
Objective Expressions	Use expressions in objectives, not assignments (no :=) or auxiliary variables.
Non-Linear Functions	Use auxiliary variables for min(), max(), or abs() in linear models.
Summation Scope	Keep indexed variables inside summation scope.
Mod Operator	Use mod as an infix operator, not a function.
Variable Bounds	Specify well-defined bounds for variables to prevent unbounded solutions.
Index Domain Matching	Ensure .dat file data aligns with .mod file index sets and parameters.
Zero-Based Indexing	Avoid index 0 unless explicitly defined in sets.
Parameter Assignments	Ensure complete parameter assignments, avoiding lone semicolons.

Figure 7 presents the outline of the prompts used at step 3 of EXEOS for generation optimization model in Python. The complete prompts are available in our replication package [3].

I. Generating Model Instruction: Generate a complete and executable Python code that accurately formulates the given structured optimization problem using Gurobi, following the provided guidelines.
II. {Gurobi Guideline}
III. {Few-shot examples}
IV. {Structured optimization problem}
V. The previous Python code for the following optimization problem failed to run. Using the solver feedback, locate the lines in the model most likely causing the errors and explain each error message. Regenerate corrected Python code that fixes all issues and follows the guidelines.
VI. {Previously Generated Optimization Model}
VII. {Solver Feedback}

Fig. 7. Prompt outline for generating an optimization model in Python in Step 3 of EXEOS: When Step 3 is invoked immediately after Step 1 for initial model creation, it uses items **I**, **II**, **III** and **IV** as its prompt. When Step 3 is invoked after Step 4 for model refinement, all items are used in the prompt.

REFERENCES

[1] AMPL. 2025. All Solvers for AMPL. https://ampl.com/products/solvers/all-solvers-for-ampl/. Accessed: 2025-04-21.
[2] AMPL. 2025. AMPL. https://ampl.com/. Accessed: 2025-09-02.
[3] Negin Ayoughi. 2025. Models or Code? Evaluating the Quality of LLM-Generated Specifications for Mathematical Optimization. GitHub repository. Artifacts and Supplementary Material [Online]. Available: https://github.com/neayoughi/EXEOS.git.