Research Article *Open Access*

Journal of Pioneering Artificial Intelligence Research

ISSN: 3069-0846

DOI: doi.org/10.63721/26JPAIR0152

The Algorithm Procurement Problem in Artificial Intelligence

Huseyin Murat Cekirge

Department of Mechanical Engineering, The City College of New York (CUNY), New York, USA

Citation: *Huseyin Murat Cekirge (2026) The Algorithm Procurement Problem in Artificial Intelligence. J of Poin Artf Research 2(4), 1-25. WMJ-JPAIR-152*

Abstract

Artificial intelligence systems can often solve the same task using substantially different computational methods. Depending on the problem, these may include rule-based and elimination procedures, lookup methods, direct algebraic solutions, pseudoinverse and regularized methods, statistical classifiers, and iterative optimization methods such as gradient descent (GD), stochastic gradient descent (SGD), conjugate gradient (CG), and Adam. The existence of alternative algorithms raises an engineering question that is distinct from whether a method can solve the task: how should an AI system select among several adequate computational methods?

This paper introduces the Algorithm Procurement Problem (APP), in which alternative computational methods are treated as competing candidates for an AI task. APP does not prescribe a particular algorithm and does not favor iterative or non-iterative computation. Instead, it imposes an algorithm-selection procedure. Candidate methods are generated, their adequacy for the requested task is evaluated, inadequate candidates are rejected, and the computational prices of the remaining candidates are compared before the task is awarded.

Computational price is defined by the application rather than by a universal measure. It may include setup and training computation, iterations, inference cost, memory and data movement, hardware utilization, energy, latency, reliability, numerical stability, repeated-use cost, and, where relevant, mission or operational requirements. Consequently, the least-cost method is not necessarily the method requiring the fewest operations or the least energy; it is the adequate method providing the best justified computational choice under the requirements of the task.

The motivation for APP is examined through exploratory ordinary-user experiments in which AI systems are allowed to select their own computational methods without an algorithm being prescribed. The experiments indicate that computationally economical alternatives may already be available to the AI system but may become explicit only when computational price is introduced into the selection request. This observation motivates a distinction between knowing an algorithm and selecting it.

APP therefore proposes that computational economy and operational requirements should participate in algorithm selection rather than being considered only after a computational paradigm has been selected. An iterative method may win when it provides the best justified solution, while a direct, lookup, rule-based, or elimination

method may win when it adequately performs the same task at a lower justified computational price.

The central proposition is that the algorithm should be treated as a bidder, while the selection procedure serves as the procurement authority.

***Corresponding author**: Huseyin Murat Cekirge, Department of Mechanical Engineering, The City Col¬lege of New York (CUNY), New York, USA. E-mail: hmcekirge@usa.net

Submitted: 18-08-2026 **Accepted**: 24-08-2026 **Published**: 31-08-2026

Abbreviations

Abbreviations	Definition
AI	Artificial Intelligence
APP	Algorithm Procurement Problem
GD	Gradient Descent
SGD	Stochastic Gradient Descent
CG	Conjugate Gradient
SVD	Singular Value Decomposition
QR	QR Decomposition

Introduction

Artificial intelligence has produced remarkable advances in prediction, classification, recognition, language processing, scientific computation, and autonomous decision-making. Much of modern AI development has consequently focused on improving model capability and predictive performance. At the same time, the computational requirements of AI systems have increased substantially, making computational efficiency an increasingly important engineering consideration [1].

The existence of multiple algorithms capable of solving the same computational problem is not new. The algorithm selection problem was formally discussed by Rice, who considered the selection of an appropriate algorithm according to the characteristics of a problem and the expected performance of available methods [2]. More recent algorithm-selection frameworks and benchmark libraries have extended this concept to a wide range of computational problems [3]. Automated machine learning has similarly addressed the selection of learning algorithms and their hyperparameters. Auto-WEKA, for example, formulates combined algorithm selection and hyperparameter optimization as an automated search over alternative learning methods and configurations [4].

Computational cost has also become an explicit concern in contemporary AI research. Green AI has emphasized computational efficiency, economic accessibility, and the reporting of computational resources in addition to predictive performance [1]. Hardware-aware neural architecture search and related approaches have further demonstrated that model architectures can be adapted to latency and resource requirements [5]. Once-for-All networks, for example, separate network training from subsequent architecture specialization so that subnetworks can be selected for different hardware platforms and resource constraints without complete retraining [6].

These developments establish that algorithm selection and computational efficiency are recognized problems. They nevertheless leave a more elementary engineering question:

At what point should computational price enter the selection procedure?

A common workflow is to select a conventional or statistically appropriate learning architecture, establish its performance, and subsequently optimize its computational implementation. Such a procedure is reasonable when feasibility is unknown. However, it can also produce post-selection optimization: computational price is investigated only after a computational paradigm has already been selected. The present work considers a different formulation. It asks whether computational price should participate in the selection of the computational method itself.

Here, computational price is not restricted to monetary

cost, FLOPs, execution time, or energy consumption. Its definition depends on the task. For a repeatedly executed commercial computation, relevant quantities may include training cost, inference cost, memory, data movement, hardware utilization, energy, and number of executions. For a real-time or safety-critical system, latency, reliability, robustness, available hardware, and operational requirements may dominate the procurement decision. There is therefore no universal definition of computational price. The relevant price must be defined by the application.

This distinction prevents the selection problem from becoming an ideological comparison between iterative and non-iterative computation. The proposed principle does not assume that deterministic, closed-form, rule-based, lookup-based, statistical, or iterative methods are inherently superior. Instead, it requires reasonable computational alternatives to compete under the requirements of the task.

The contribution proposed here is therefore not another competing learning algorithm. It is a procedure governing competition among available algorithms.

For a given AI task, candidate solutions may belong to fundamentally different computational paradigms. Depending on the problem, these may include feature or candidate elimination, deterministic rules, lookup procedures, direct algebraic solutions, statistical classifiers, iterative numerical methods, gradient-based optimization, or neural-network training. Existing automated approaches can search substantial spaces of learning algorithms and hyperparameters, but such searches may themselves be computationally expensive and are generally formulated within predefined model, pipeline, or optimization spaces [4,7].

The Algorithm Procurement Problem (APP) begins from a simpler engineering principle:
Do not impose an algorithm. Impose an algorithm-selection procedure.

The AI system should first identify reasonable computational methods capable of performing the requested task. Methods that cannot provide acceptable accuracy, stability, robustness, latency, safety, or other required operational characteristics are rejected. Among the remaining adequate candidates, computational price becomes an explicit selection criterion.

An iterative method such as GD, SGD, CG, or Adam is therefore neither prohibited nor preferred. If an iterative method provides the best justified computational solution, it should be selected. Conversely, when elimination, lookup, direct algebra, or another procedure provides adequate performance at a lower justified computational price, there is no engineering reason to impose iteration merely because it represents conventional practice for the application.

The proposed procurement sequence is therefore:
AI TASK
↓
MISSION AND PERFORMANCE REQUIREMENTS
↓
CANDIDATE COMPUTATIONAL METHODS
↓
ADEQUACY EVALUATION
↓
COMPUTATIONAL PRICE EVALUATION
↓
METHOD SELECTION

An important motivation for this work arose from simple AI-assisted programming experiments. When small numerical problems were presented without prescribing an optimization method, direct least-squares and pseudoinverse-based solutions could be selected without resorting to gradient-based iteration. When a small recognition problem was instead framed as a conventional binary-classification task, an iterative regularized Logistic Regression procedure could initially be recommended. Explicitly asking whether iteration was computationally necessary caused lower-cost non-iterative alternatives to enter consideration.

These observations do not establish a general behavioral property of contemporary AI systems.

They instead motivate an experimentally testable question:
How and why does an AI system select one computational method when several methods capable of solving the same task are available?

A second question follows directly:
Does the selected method change when computational price, latency, energy, robustness, reliability, or mission requirements are introduced explicitly into the same problem?

The objective of this paper is therefore not to demonstrate that iterative optimization is inefficient, nor to argue that direct methods should replace modern machine learning. It is to investigate whether computational and operational price should become an explicit and systematic component of AI algorithm selection itself.

In procurement terminology, algorithms are competing bidders for a computational task. The algorithm is not the procurement authority. The selection procedure is the procurement authority. A further issue arises before alternative computational methods can be compared. The criteria governing their comparison may not be completely specified by the user. When an AI system receives an underspecified task, it may interpret the user's intent and infer additional considerations such as flexibility, extensibility, simplicity, computational efficiency, or anticipated future use. Such inference is neither unexpected nor necessarily undesirable; it is part of producing a useful response from an incomplete specification. However, the inferred criteria may influence which computational methods enter the candidate set and which method is ultimately recommended. Algorithm procurement therefore involves not only the comparison of candidate algorithms, but also recognition of the criteria under which that comparison is being made.

Definition of the Algorithm Procurement Problem

An artificial intelligence task can often be performed using more than one computational method. Depending on the structure and requirements of the problem, possible candidates may include gradient descent (GD), stochastic gradient descent (SGD), Adam, conjugate gradient (CG), direct algebraic methods, pseudoinverse or regularized solutions, Naive Bayes, lookup procedures, rule-based methods, elimination-based methods, and other applicable computational procedures.

The Algorithm Procurement Problem (APP) begins with a simple principle:

No candidate algorithm receives automatic priority.

The objective is not to assume in advance that an iterative, non-iterative, statistical, algebraic, rule-based, or other computational method should be used. Instead, reasonable alternative methods are treated as candidates for performing the same AI task.

The central principle of APP is:

We do not impose an algorithm. We impose an algorithm-selection procedure.

Under this interpretation, the algorithm is not the decision maker. It is a candidate competing to perform a computational task.

The algorithm is a bidder.

The AI task therefore becomes analogous to an engineering procurement problem. A computational requirement exists, several methods may potentially satisfy that requirement, and a procedure is required to determine which candidate provides the best justified solution under the requirements of the application.

Candidate methods may include:
GD
SGD
Adam
CG
Direct algebra
Pseudoinverse / Ridge
Naive Bayes
Lookup methods
Rule-based methods
Elimination-based methods
Other applicable computational procedures

This list is neither exhaustive nor ordered. Its ordering does not imply computational preference. A direct method is not automatically preferred to an iterative method, nor is an iterative method automatically preferred because it represents common practice in machine learning.

Each candidate must first demonstrate that it can adequately perform the requested task. Adequacy may include accuracy, numerical stability, robustness,

reliability, latency, safety, scalability, or other requirements imposed by the application. Candidates that fail the required adequacy conditions are rejected before computational price is compared.

APP therefore separates two questions that are frequently combined:

Question 1: Can the method adequately satisfy the requirements of the AI task?

Question 2: What is the computational price of using that method under the intended operating conditions?

A computationally inexpensive method that cannot adequately satisfy the task requirements is not a valid candidate. Conversely, once several methods satisfy the required conditions, selection of a substantially more expensive method requires an engineering justification.

Computational price is not a universal quantity and is not restricted to execution time, monetary cost, energy consumption, or FLOP count. Its definition depends on the application. Relevant components may include training computation, number of iterations, preprocessing, repeated inference operations, memory requirements, data movement, hardware utilization, energy consumption, latency, reliability, repeated-use cost, and other operational or mission requirements.

For example, in a large-scale repeatedly executed application, energy, memory, hardware utilization, and the number of executions may dominate the computational price. In a real-time or safety-critical application, latency, reliability, robustness, available computational resources, and mission requirements may dominate instead.

Thus, APP does not necessarily seek the method requiring the least computation. It seeks the adequate method providing the **lowest justified computational price as defined by the application**.

The procurement problem can therefore be represented conceptually as:

AI TASK
↓
TASK AND OPERATIONAL REQUIREMENTS
↓
CANDIDATE COMPUTATIONAL METHODS
↓
ADEQUACY EVALUATION
↓
REJECTION OF INADEQUATE METHODS
↓
COMPUTATIONAL PRICE EVALUATION
↓
METHOD SELECTION

An important feature of this formulation is that the user is not required to know the available algorithms in advance. A user requesting an AI solution should not need to specify GD, SGD, Adam, pseudoinverse, Ridge regression, lookup, elimination, or another particular computational procedure merely to obtain an appropriate and economical solution. Identifying reasonable alternatives should itself be part of the algorithm-selection process.

Thus, APP does not propose a new competitor to existing AI algorithms. It proposes a procedure under which existing and future algorithms compete.

The distinction can be summarized as follows:

The contribution is not a new bidder. It is a bidding procedure.

Under APP, GD may win. Adam may win. A direct algebraic method may win. A lookup table, statistical classifier, rule-based method, or elimination procedure may win. The outcome is not prescribed beforehand.

The fundamental requirement is that a computational method should not receive the task merely because it is conventional, familiar, or commonly associated with a particular class of AI problems. Nor should a computationally cheaper method receive the task merely because it is cheaper.

The method must satisfy the task requirements and then compete on the computational price relevant to the application.

The algorithm is a bidder.
The selection procedure is the procurement authority.

The Algorithm Procurement Procedure

The Algorithm Procurement Problem requires a systematic procedure for selecting a computational method before a particular computational paradigm is adopted. The purpose of the procedure is not to favor a specific class of algorithms, but to require reasonable alternatives to be considered before the computational task is awarded.

The proposed Algorithm Procurement Procedure consists of six steps.

Step 1: Define the AI Task and Its Requirements

The computational task must first be defined independently of the algorithm that will eventually be used to solve it.

At this stage, the problem is described in terms of the required inputs, outputs, expected performance, and relevant operational requirements. Depending on the application, these requirements may include accuracy, numerical stability, robustness, reliability, latency, memory limitations, available hardware, energy constraints, safety, or other mission-specific conditions.

No assumption is made that the task requires gradient descent, stochastic gradient descent, neural-network training, direct algebra, or any other particular computational procedure.

This separation is fundamental. Defining what the task must accomplish and selecting how it should be accomplished are two different engineering decisions.

Step 2: Generate Candidate Computational Methods

Once the task and its requirements have been defined, reasonable alternative computational methods are identified.

Depending on the problem, candidates may include:
GD
SGD
Adam
CG
Direct algebraic methods
Pseudoinverse methods
Regularized direct methods
Naive Bayes
Lookup methods
Rule-based methods
Elimination-based methods
Other applicable computational procedures

The candidate list is not ordered according to preference. No method receives priority merely because it is commonly associated with a particular type of AI problem.

The purpose of this step is to prevent premature commitment to a computational paradigm.

The candidate-generation procedure should itself remain economical. APP does not require an exhaustive search over every conceivable algorithm; it requires reasonable and applicable alternatives to be considered.

Step 3: Test Whether Each Candidate Adequately Satisfies the Task Requirements

Each candidate method is evaluated to determine whether it can adequately satisfy the requirements established in Step 1.

Adequacy may be determined from prediction accuracy, classification performance, numerical stability, robustness, reliability, latency, safety, scalability, or other measures appropriate to the application.
The important principle is that computational price alone cannot qualify a method.

A method that is inexpensive but cannot satisfy the required operating conditions is not a valid bidder.
For a safety-critical or real-time application, for example, a computationally inexpensive method that fails the required latency or reliability constraint must be rejected regardless of its lower energy or execution cost.

Step 4: Reject Inadequate Candidates

Candidate methods that fail the adequacy evaluation are removed from further consideration.

This produces a reduced set of admissible computational methods.

If a simple rule-based procedure cannot adequately solve the problem, it is rejected.

If a closed-form method cannot provide adequate performance, it is rejected.

If an iterative method cannot provide adequate performance, it is also rejected.

The rejection procedure is independent of computational paradigm.

No candidate is protected because it is conventional, and no candidate is rejected merely because it is iterative or non-iterative.

Only candidates satisfying the required task and operational conditions proceed to computational-price evaluation.

Step 5: Determine the Computational Price of the Remaining Candidates

The computational price of each remaining candidate is then evaluated.

Computational price is application-dependent and may include:
Training computation
Number of iterations
Preprocessing computation
Memory requirements
Data movement
Hardware requirements
Energy consumption
Inference computation
Latency
Repeated-use cost
Reliability-related computational requirements
Other application-specific operational costs

The relevant components and their relative importance depend on the application.

For a method that is trained once and executed millions of times, inference cost may dominate the total computational price.

For another problem, training cost may dominate.

A method requiring substantial precomputation may still be economical when the resulting operator can subsequently be reused for a large number of evaluations.

In a real-time or safety-critical system, latency or reliability may dominate energy or monetary cost. In a large-scale repeatedly executed application, relatively small differences in energy, memory movement, or inference computation may become important because they are accumulated over millions or billions of executions.

Therefore, computational price should be evaluated over the expected operational life and operating conditions of the method rather than from a single isolated computation.

Step 6: Award the Task to the Best Justified Adequate Candidate

After inadequate methods have been rejected and the computational prices of the remaining candidates have been evaluated, the task is awarded to the candidate providing the best justified computational choice under the requirements of the application.

This is not necessarily the candidate requiring the fewest operations, the least energy, the lowest monetary expenditure, or the shortest execution time considered individually.

The winning candidate is the method that satisfies the required conditions while providing the most appropriate computational price under the priorities of the application.

The procedure can therefore be summarized as:
AI TASK
↓
TASK AND OPERATIONAL REQUIREMENTS
↓
CANDIDATE COMPUTATIONAL METHODS
↓
ADEQUACY EVALUATION
↓
REJECTION OF INADEQUATE CANDIDATES
↓
APPLICATION-DEPENDENT COMPUTATIONAL PRICE EVALUATION
↓

AWARD

The resulting selection is not predetermined.

An iterative optimization method may win.

A direct algebraic method may win.

A lookup procedure may win.

A rule-based or elimination-based procedure may win. The winning method depends on the task, its ability to satisfy the required operating conditions, and its computational price as defined for the application.

This distinguishes APP from an approach in which an algorithm is selected first and its computational efficiency is investigated afterward.

The Algorithm Procurement Procedure requires reasonable computational alternatives to compete before the computational method is selected.

The governing principle is therefore:
We do not impose an algorithm. We impose an algorithm-selection procedure.

The algorithm is a bidder.

The task should be awarded only after the bids have been evaluated.

The Ordinary-Citizen Experiment
The motivation for the Algorithm Procurement Problem emerged from a simple question:

What computational method will an artificial intelligence system select when an ordinary user requests an AI solution without specifying how the computation should be performed?

The purpose of the experiment was not initially to compare particular numerical algorithms. Instead, the intention was to observe the method-selection behavior of the AI system when the user supplied the problem but did not prescribe GD, SGD, Adam, direct algebra, pseudoinverse, or another computational procedure.

The user was therefore treated as an ordinary citizen with a computational problem rather than as an expert capable of specifying the solution method.
The basic experimental principle was:
Give the AI the task

Do not specify the algorithm

Observe the algorithm selected

Then change or clarify the selection requirements and observe whether the selected computational method changes

This final step became increasingly important as the experiment progressed. The initial question concerned computational economy, but the broader issue became algorithm-selection behavior itself. The experiment therefore examined not only whether a lower-cost method could be found, but whether explicitly introducing computational price, latency, stability, repeated use, or other requirements changed the set of methods considered by the AI system.

The experiment consequently evolved from a simple test of computational efficiency into a broader question:

How and why does an AI system select one computational method when several methods capable of performing the same task are available?

Initial 8 × 5 Problem
The first experiment consisted of eight training examples with five numerical input features and one numerical output.

The problem was deliberately presented as an artificial intelligence problem. The user stated that the objective was to build a small AI model capable of learning the relationship between the five inputs and the output. No optimization algorithm was specified.

The initial suspicion was that framing the problem as an AI training task might automatically lead to an iterative learning procedure such as GD, SGD, or Adam.
This did not occur.

The AI system recognized the mathematical structure as a small linear least-squares problem and selected the NumPy least-squares solver:

np.linalg.lstsq(X, y, rcond=None)

Thus, although the problem had been presented as an AI problem, the selected computational procedure was a numerical least-squares solution rather than an explicitly gradient-based training loop. This result was important because it contradicted the initial suspicion that an AI system would necessarily associate an AI training problem with iterative optimization.

The experiment therefore changed direction.

The question was no longer:
Will the AI automatically select an iterative training method?

Instead, a broader question emerged:
How and why does the AI system select a particular computational method from the alternatives available to it?

This change in direction became an important part of the experiment itself.

Repeated Computation with a Fixed Matrix

The next experiment introduced repeated use.

The feature matrix remained fixed while a large number of target vectors were to be processed.

The AI system recognized that repeatedly solving the complete least-squares problem could perform unnecessary repeated computation. It proposed computing the Moore-Penrose pseudoinverse once and reusing it:
$A^+ = pinv(A)$

followed by

$w = A^+b$

for each new target vector.

This result demonstrated that algorithm selection depended not only on the mathematical form of a single computation but also on the expected pattern of use.

A method appropriate for one execution is not necessarily the most appropriate method when the same structural problem is executed repeatedly. Precomputation changes the procurement price.

Computational price must therefore be evaluated over the expected operational use of the method rather than solely from the cost of obtaining one solution.

Underdetermined Problem

The dimensions were subsequently reversed to produce a system containing five examples and eight parameters.

The AI system identified the problem as underdetermined and selected a minimum-norm solution using the Moore-Penrose pseudoinverse.

For an appropriate full-row-rank matrix A, the solution can be represented as

$w = A^+b$

with

$A^+ = A^T(AA^T)^{-1}$.

Again, no iterative optimizer was introduced merely because the problem had been presented in an AI context. The system adapted the computational procedure to the mathematical structure of the problem. This provided a second indication that algorithm selection was responsive to problem structure rather than being determined solely by the label “artificial intelligence.”

Perturbation and Regularization

The experiment was then modified by introducing small perturbations into the input matrix. The purpose was to determine whether the AI system would recognize that computational price alone was insufficient when numerical stability became important.

The system proposed regularization and produced a regularized solution of the form

$w = A^T(AA^T + \lambda I)^{-1}b$

for the underdetermined case considered. This result reinforced an important feature of the procurement concept. Computational economy cannot be evaluated independently of adequacy. A computationally

inexpensive method that becomes numerically unstable under the operating conditions of the problem may cease to be an admissible bidder. The experiment therefore introduced another selection requirement:
A method must remain adequate under the conditions in which it will be used.

Transition from Linear Algebra to an AI Recognition Task

The preceding experiments remained closely connected to linear algebra. A different experiment was therefore introduced to observe algorithm-selection behavior when the problem was described as a more conventional AI recognition task.

The user asked for a system capable of recognizing a dog using several observable features.

Some features were directly related to the target, such as:
dog shape
four legs
tail
collar
barking indication

Other observations represented contextual or potentially irrelevant information, including:
person
glasses
ring
pants
shoes
chair
table
grass
door

Some of these observations were deliberately described as strongly related or repeatedly occurring together. For example, a ring could repeatedly occur together with a person and pants.

The AI system first proposed reducing the information supplied to the classifier. It recommended removing irrelevant and redundant features before training.

This was significant because computational reduction occurred before model training.

The resulting sequence was approximately:

INPUT INFORMATION
↓
FEATURE EVALUATION
↓
ELIMINATION OF IRRELEVANT OR REDUNDANT INFORMATION
↓
REDUCED FEATURE SET
↓
MODEL TRAINING

The system then selected regularized Logistic Regression for the reduced binary-classification problem.

Unlike the earlier least-squares examples, this choice introduced an iterative optimization procedure. This result was also important. The same AI system that had selected non-gradient numerical procedures for the preceding problems selected an iterative procedure when the task was presented as a conventional binary-classification problem. The experimental question therefore became increasingly concerned with the behavior of algorithm selection itself.

The Computational-Price Question

At this point, the user explicitly questioned the computational procedure. The question was not whether Logistic Regression was mathematically valid. It was a technically reasonable method for the stated classification problem. The question was instead:

Was iterative optimization necessary for this particular small reduced problem, or could another adequate computational method perform the same task at a different computational price?

The AI system was therefore asked to reconsider the problem while explicitly including computational price in the selection criteria. The response changed. The system proposed several alternatives, including:

Naive Bayes based on counting or lookup
Closed-form linear or regularized classification
Rule-based or threshold-based classification.

For the particular illustrative example, the system ultimately proposed a closed-form regularized linear alternative. This produced a central observation of the ordinary-citizen experiment. The alternative computational methods were not unknown to the AI system. They entered the discussion when the selection requirements changed. This observation suggests a distinction between two different questions:
What algorithms does the AI system know?

And

Which of those algorithms does the AI system bring into consideration for a particular task under a particular set of requirements?

Why Did The Candidate Set Change?
A follow-up question examined why the alternatives had not been proposed during the initial recognition problem.

The initial recommendation had followed a conventional machine-learning interpretation of the task: binary classification naturally suggested Logistic Regression, while computational price had not been explicitly established as a primary selection requirement.

When computational price was subsequently introduced, the candidate set and the selection logic changed. This distinction is central to the present study. Knowledge of an alternative computational method does not necessarily imply that the method will enter the initial algorithm-selection process. The issue is therefore not simply whether an AI system has knowledge of GD, SGD, Adam, direct algebra, pseudoinverse, regularization, Naive Bayes, lookup procedures, or rule-based methods.

The more fundamental issue is:
What causes a particular method to enter or leave the candidate set?

The experiment suggests that algorithm selection may depend not only on the mathematical task but also on how the requirements of that task are expressed.

Computational price is one such requirement. Numerical stability, latency, reliability, robustness, repeated use, hardware limitations, or mission requirements may be others.

The Algorithm Procurement Problem therefore concerns not merely the search for a cheaper algorithm, but the procedure by which computational alternatives are identified, evaluated, rejected, and selected.

Interpretation of the Pilot Experiment

The ordinary-citizen experiment should not be interpreted as proof that contemporary AI systems systematically select unnecessarily expensive algorithms.

Indeed, the initial 8 × 5 experiment demonstrated the opposite of the original suspicion: when the mathematical structure supported a least-squares formulation, the AI system selected an appropriate numerical least-squares procedure without being instructed to avoid iterative optimization. The observations are exploratory and require controlled repetition across different AI systems, prompts, problem classes, and computational environments. Nevertheless, the experiment reveals a broader and experimentally testable research question:

How and why does an AI system select one computational method when several methods capable of adequately performing the same task are available?

A related experimental question is:
Does the selected method change when the requirements supplied to the AI system change?

This can be investigated by comparing controlled conditions.

Condition A:
The AI system receives the task and is asked to choose an appropriate computational method.

Condition B:
The AI system receives the same task but is additionally required to evaluate reasonable alternative methods according to explicitly stated criteria such as computational price, latency, numerical stability, energy, repeated-use cost, robustness, reliability, or other application requirements.

The resulting candidate sets and selected methods can then be compared. Where executable implementations are available, training cost, inference cost, number of iterations, memory requirements, latency, energy, and other relevant quantities can also be measured or estimated under stated assumptions.
The purpose is not to force the AI system toward a non-iterative solution.
If an iterative method provides the best justified solution under the applicable requirements, it should be selected.

If a direct, lookup, rule-based, elimination-based, statistical, or other method provides the better justified solution, that method should be selected instead.

The ordinary-citizen experiment therefore leads to a broader hypothesis than the one initially considered:

The user should not be required to know which algorithms should compete.

Identifying reasonable computational alternatives, evaluating their adequacy, and selecting among them according to the requirements of the application should itself be part of the algorithm-selection procedure.

The central experimental question is therefore no longer simply whether a cheaper algorithm exists.

It is:
How and why does an AI system select an algorithm?

When the AI Becomes Selection-Aware

The preceding experiments produced an unexpected transition. The AI system did not appear to lack knowledge of alternative computational methods. Rather, different alternatives became prominent when the requirements used for algorithm selection were changed or made explicit.

This distinction is important to the Algorithm Procurement Problem. The question is not simply:
Does the AI know another algorithm?

The more relevant question is:
Which algorithms does the AI bring into consideration for a particular task, and what causes the candidate set or final selection to change?

Computational price is one important selection criterion, but it is not the only one. Accuracy, numerical stability, latency, reliability, robustness, energy, hardware limitations, repeated use, and mission requirements may also influence which computational method provides the best justified solution.

The Initial Selection

In the recognition experiment, the AI system first reduced the feature space by eliminating irrelevant and redundant information. This demonstrated that the system was capable of computational simplification before training.

After reducing the problem, the system selected Logistic Regression with L2 regularization.

This was a conventional and technically reasonable choice for binary classification. The selected implementation nevertheless required iterative optimization.

At this stage, computational price had not been explicitly requested by the user.

The procedure was approximately:
AI TASK
↓
FEATURE REDUCTION
↓
STANDARD MODEL SELECTION
↓
LOGISTIC REGRESSION
↓
ITERATIVE TRAINING

The important observation is not that this sequence was incorrect. The important observation is that alternative computational paradigms were not initially compared under an explicit computational-price criterion. This contrasts with the preceding numerical experiments, in which the AI system had selected least-squares and pseudoinverse-based procedures without being instructed to avoid iteration. The experiments therefore did not reveal a universal preference for iterative computation. They revealed that algorithm selection changed with the structure and framing of the task.

The Price Question

The user then introduced an additional question:
Is iterative optimization actually necessary for this particular problem?

The AI system was asked whether the reduced recognition problem could be solved adequately using alternative computational procedures and whether those alternatives could offer a different computational price. This apparently small change in the request produced a significant change in the response. The AI system reconsidered the computational alternatives and proposed:
Naive Bayes
Lookup and counting procedures
Closed-form linear classification
Regularized direct algebraic solutions
Rule-based and threshold-based classification

The important observation was not that these methods were necessarily superior to Logistic Regression. Rather, these alternatives had not been prominent in the initial response but entered the candidate set after computational price was explicitly introduced. The issue was therefore not simply the availability of algorithmic knowledge.

The issue was **when and why that knowledge entered the selection procedure.**

The AI Explains its Initial Decision

The AI system was subsequently asked why it had initially recommended regularized Logistic Regression rather than the alternatives that it later identified. The system explained its initial recommendation in terms of the conventional interpretation of the problem. Binary classification suggested Logistic Regression, L2 regularization provided stability, and an iterative solver represented a standard implementation. Computational price had not initially been established as an explicit selection criterion. When computational price was introduced, the selection process broadened to include alternative computational paradigms. The sequence can therefore be represented as:

Initial request:
TASK
↓
STANDARD AI/MACHINE-LEARNING INTERPRETATION
↓
CONVENTIONAL MODEL SELECTION
↓
TRAINING

After the price question:
TASK
↓
EXPLICIT SELECTION REQUIREMENTS
↓
ALTERNATIVE COMPUTATIONAL METHODS
↓
ADEQUACY EVALUATION
↓
COMPUTATIONAL PRICE EVALUATION
↓
SELECTION

This difference provides an important motivation for APP.

The issue is not whether conventional model selection is inherently incorrect. The issue is whether reasonable alternatives should be considered before a computational paradigm receives the task.

Knowing is not Selecting

An AI system may know many computational methods without automatically comparing all of them for a particular task.

Knowledge of GD does not require selection of GD.

Knowledge of Adam does not require selection of Adam.

Knowledge of a pseudoinverse does not require selection of a pseudoinverse.

Knowledge of a lookup procedure does not require selection of a lookup procedure.

Algorithmic knowledge and algorithmic selection are different operations.

Therefore:

Knowing An Algorithm ≠ Selecting The Algorithm

and, more generally,

Knowing An Alternative ≠ Including That Alternative In The Candidate Set

The experiments suggest that the computational methods brought into consideration may depend on the task description and on the selection requirements explicitly or implicitly associated with that task.

Computational price is one such requirement.
The procurement logic can therefore be illustrated as:
Rule → inadequate under required conditions → **REJECT**
Ridge → adequate + best justified computational choice → **AWARD**
Adam → adequate + best justified computational choice → **AWARD**

The winner is not determined by computational paradigm.

The problem is therefore not necessarily that the AI system is incapable of producing an economical or operationally appropriate solution.

The question is whether the relevant alternatives and selection criteria enter the procurement procedure before the computational task is awarded.

The Ordinary User Should Not Need to Select the Algorithm

This issue becomes particularly important for a non-specialist user.

An ordinary user may know the task but may not know:
GD
SGD
Adam
CG
SVD
QR decomposition
Pseudoinverse
Ridge regularization
Naive Bayes
Lookup methods
or other computational alternatives.

It would therefore be unreasonable to require the user to identify the appropriate algorithmic alternatives by name.

For example, a user should not need to say:
"Do not use iterative Logistic Regression. First investigate whether a closed-form regularized classifier can provide equivalent adequate performance."
Such a request already assumes substantial knowledge of machine learning and numerical analysis. Instead, the user should be able to describe the objective and the relevant requirements in ordinary engineering language:

Perform the task adequately and avoid unnecessary computational expenditure.

For another application, the instruction might instead emphasize latency, reliability, memory, energy, or another operational requirement. The AI system should then identify reasonable computational alternatives and evaluate them under those requirements. This is one of the principal motivations for imposing an algorithm-selection procedure rather than requiring the user to impose a particular algorithm.

The Post-Selection Problem

A conventional workflow may be represented as:
SELECT METHOD
↓
TRAIN OR IMPLEMENT METHOD
↓
MEASURE PERFORMANCE
↓
MEASURE COMPUTATIONAL COST
↓
OPTIMIZE COST IF NECESSARY

In such a workflow, computational price may be evaluated after an important method-selection decision has already been made.

APP asks whether relevant computational considerations should enter earlier:
DEFINE TASK AND REQUIREMENTS
↓
GENERATE REASONABLE ALTERNATIVE METHODS
↓

DETERMINE ADEQUATE CANDIDATES
↓
COMPARE APPLICATION-DEPENDENT COMPUTATIONAL PRICE
↓
SELECT METHOD
↓
EXECUTE OR TRAIN

This is not necessarily premature optimization.

APP does not require exhaustive optimization of every possible implementation before the problem is understood.

Instead, computational price is considered as one component of algorithm selection itself.

The distinction is between:
Post-selection optimization
and
Pre-selection procurement

The Procurement Authority

The central conceptual distinction can now be stated more precisely.

No computational method should determine its own entitlement to the task.

A neural network is a bidder.
Gradient descent is a bidder.

Adam is a bidder.

A direct algebraic method is a bidder.

A lookup procedure is a bidder.

An elimination procedure is a bidder.

None is the procurement authority.

The procurement authority is the selection procedure.

Accordingly:
The algorithm is a bidder.

The selection procedure is the procurement authority.

The purpose of this principle is to prevent convention alone, or any predetermined preference for a computational paradigm, from determining the selected method.

When The Iterative Method Should Win

APP is not intended to eliminate iterative optimization.

There will be problems for which rules cannot provide adequate performance.

There will be problems for which lookup methods are impractical.

There will be problems for which no useful direct solution exists.
There will also be nonlinear, high-dimensional, large-scale, or changing problems for which iterative optimization provides the best justified computational choice.

In such cases, the iterative method should win.

The important difference is that it wins after evaluation rather than through automatic preference.

APP therefore does not state:
Use non-iterative methods.

It states:
Compare reasonable adequate methods before selection.

The same principle applies in the opposite direction. A direct or non-iterative method should not win merely because it avoids iteration. It must satisfy the requirements of the task and provide a justified computational choice.

From Default Selection to Procurement

The experiments suggest a transition from default algorithm selection toward explicit algorithm procurement.

The conventional question may be:
Which algorithm is normally used for this type of AI problem?

The procurement question is:
Which available computational method provides the best justified solution for this particular task under its actual requirements?
These questions can produce the same answer.

They can also produce different answers.
The distinction becomes particularly important when AI systems generate code or computational solutions for users who may have little or no knowledge of the algorithms operating underneath the generated program.

A technically correct program may still contain unnecessary computation.

Conversely, the computationally cheapest program may fail important requirements such as robustness, latency, numerical stability, reliability, or safety. Correctness alone does not establish computational economy, and computational economy alone does not establish adequacy. APP requires both to participate in justified method selection.

Central Observation

The experiments presented in this work are preliminary and do not establish that AI systems systematically favor expensive algorithms, iterative algorithms, or any other computational paradigm. They support a narrower and experimentally testable observation:

Changing or explicitly stating the selection requirements can change the set of algorithms considered and can change the computational method recommended by the AI system.

Computational price provides one demonstrated example in the exploratory experiments. This leads to the principal hypothesis of APP:

Algorithm selection should explicitly consider the requirements that determine the computational value of competing methods before a computational paradigm is adopted.

The resulting principle can be stated as:
Do not select first and ask the price later.

Include the price in the selection procedure.

Here, "price" is defined by the application. It may include computation, latency, memory, energy, reliability, repeated use, hardware requirements, or other relevant operational considerations.

Under this formulation, the AI system is free to propose any applicable computational method.

The method selected, however, should be justifiable under the requirements of the task.

The algorithm is a bidder.

The selection procedure is the procurement authority.

Who Determines the Algorithm-Selection Criteria?

The ordinary-citizen experiment revealed an additional stage preceding algorithm selection. Before candidate algorithms can be compared, the criteria governing that comparison must themselves be established.

In a subsequent ordinary-user test, the AI system was given a general request to construct a small AI model that would learn a relationship between input data and corresponding outputs and predict outputs for new inputs. The user did not specify a neural network, an optimization method, a numerical procedure, computational price, scalability, or other algorithm-selection criteria.

The AI system initially proposed a small neural network implemented in PyTorch and trained using Adam. When subsequently asked why this approach had been selected, the system identified several considerations, including flexibility, pedagogical clarity, extensibility, and computational efficiency. When asked which considerations originated from the user's request and which had been inferred, the system acknowledged that some technical considerations had been supplied through its own interpretation of the underspecified task.

This observation should not be interpreted as evidence of an incorrect selection. An underspecified request necessarily requires some interpretation, and the resulting assumptions may be reasonable and useful. The observation instead identifies an additional stage in the algorithm-selection process.

The process may therefore be represented as:
USER TASK
↓
INTERPRETATION OF USER INTENT
↓
EXPLICIT AND INFERRED SELECTION CRITERIA
↓
CANDIDATE COMPUTATIONAL METHODS
↓
ADEQUACY AND PRICE EVALUATION
↓
METHOD SELECTION

The distinction is important because different reasonable criteria can favor different computational methods. A preference for flexibility and future extensibility may favor a general-purpose learning architecture. A requirement emphasizing computational minimality, latency, repeated use, numerical stability, or another operational property may produce a different candidate set or ranking. Neither outcome is necessarily incorrect; they correspond to different procurement requirements.

This introduces a question that precedes the Algorithm Procurement Problem:
Who determines the criteria under which the algorithms compete?

Some criteria are explicitly supplied by the user. Some follow directly from the technical requirements of the task. Others may have to be inferred by the AI system because the request is incomplete.

Thus, algorithm selection can be viewed as involving three distinct decisions:

1. **Task determination — What must be accomplished?**
2. **Criteria determination — What properties should govern method selection?**
3. **Algorithm selection — Which candidate best satisfies those criteria?**

The third decision cannot always be interpreted independently of the second.

This observation does not require the AI system to avoid making assumptions. In many ordinary-user interactions, some assumptions are unavoidable. Rather, it suggests that the assumptions materially affecting algorithm selection may themselves be relevant to understanding and evaluating the resulting computational choice.

APP therefore does not claim that AI systems should be instructed which algorithm to use or which criteria to prefer. It proposes that, when algorithm selection is delegated to an AI system, the criteria influencing that selection form part of the procurement process and can themselves be examined experimentally.

The resulting question is broader than whether an AI system selects an iterative or non-iterative method:

What criteria does an AI system use when selecting a computational method, where do those criteria originate, and how do they affect the resulting selection?

Discussion and Conclusions

The Algorithm Procurement Problem was developed from a simple engineering concern: an artificial intelligence system may have access to several computational methods capable of addressing a task, but knowledge of those methods does not guarantee that they will all enter consideration before one of them is selected.

The exploratory experiments presented in this study suggest that this distinction deserves explicit consideration in AI-assisted problem solving. The computational method proposed by an AI system may depend not only on the mathematical structure of the problem but also on the requirements explicitly or implicitly included in the request.

The proposed framework does not introduce a new training algorithm. It introduces a procedure for selecting among existing and future computational alternatives. The central issue is therefore not whether one particular class of algorithms is superior. It is how and why a computational method is selected.

The Central Engineering Problem

Modern AI systems provide users with increasingly powerful capabilities for generating models, algorithms,

and executable code. A user may describe a problem without specifying the numerical, statistical, or optimization procedure to be used. This capability transfers an important engineering decision from the user to the AI system. The AI system is no longer being asked only:
How should this algorithm be implemented?

It may also implicitly be asked:
Which algorithm should be used?

These are fundamentally different questions.

When algorithm selection is delegated to the AI system, the system may have several computationally different methods capable of adequately performing the task. The selection procedure must therefore determine not only whether a method works, but whether its selection is justified under the requirements of the application.

A mathematically valid solution is not necessarily the most appropriate computational solution. Conversely, the computationally cheapest solution is not necessarily an adequate solution. APP addresses the decision between these alternatives.

Correctness, Adequacy, and Computational Economy

Correctness and adequacy remain primary requirements. APP does not propose sacrificing accuracy, stability, reliability, safety, or other required performance merely to reduce computational expenditure. Suppose two candidate methods, A and B, both satisfy the requirements of the task.

If
$C(A) > C(B)$,

where C represents the computational price relevant to the application, Method A requires additional justification for selection. Such justification may exist.

Method A may provide improved robustness, scalability, maintainability, generalization, latency, reliability, safety, or future adaptability. If these benefits are important to the application, they must participate in the procurement decision. Thus, APP does not require selection of the method with the smallest numerical value of computational cost independently of all other considerations. It requires the computational choice to be justified. The central question is therefore not simply:

Which method is cheaper?

It is:
Which adequate method provides the best justified computational choice under the requirements of this application?

Computational Price is Application-Dependent

Computational price should not automatically be identified with execution time, FLOP count, monetary expenditure, or energy consumption. There is no universal definition of computational price. For a particular task, relevant components may include:

Training Computation
Precomputation
Number Of Iterations
Inference Computation
Memory Requirements
Memory Transfers
Hardware Requirements
Energy Consumption
Latency
Reliability Requirements
Retraining Requirements
Expected Number Of Executions
and Other Application-Specific Or Mission-Specific Requirements.
The importance assigned to these quantities depends on the application.

Consider a fixed matrix A and a large sequence of changing vectors b. Repeatedly solving
$Aw \approx b$

from the beginning may be unnecessary if a reusable operator can be constructed once.

A method that appears expensive during initial preparation may therefore become economical over its operational lifetime.

For such an application, the computational price may be represented conceptually as

$C_{total} = C_{setup} + C_{training} + N_{use} C_{execution}$,

where
C_{total} = total computational price
C_{setup} = setup or preprocessing price
$C_{training}$ = training price
$C_{execution}$ = price per execution
and
N_{use} = expected number of executions

Additional terms may be introduced for memory, data movement, energy, hardware, latency, reliability, or other relevant requirements. A different application may produce an entirely different price model. For example, consider a real-time safety-critical system in which two methods provide different combinations of accuracy, latency, and energy consumption. A method with lower energy consumption may be unacceptable if its latency exceeds a mission requirement. Conversely, a method consuming more energy may be justified if it provides the required response time and reliability. For such an application, computational price may conceptually depend on

C = f(latency, reliability, accuracy, energy, hardware, mission risk).

For a large-scale commercial computation executed millions or billions of times, the relevant price may instead depend more strongly on

C = f(computation, energy, infrastructure, hardware, latency, repetitions, monetary cost).

Therefore:
The computational price must be defined by the application.

APP requires price to participate in algorithm selection; it does not require every application to use the same definition of price.

The Role of Iterative Optimization

The experiments presented in this work should not be interpreted as an argument against GD, SGD, Adam, CG, or iterative optimization in general.

Indeed, the initial numerical experiment demonstrated that the AI system did not automatically select an iterative method merely because the task had been described as an AI problem.

Iterative methods remain indispensable for many computational problems.

For complex nonlinear systems, high-dimensional models, large-scale learning problems, changing environments, and problems for which useful direct solutions are unavailable, iterative optimization may provide the best practical and computationally justified solution.

In such cases, the iterative method should receive the task.

APP changes only the basis on which that decision is made.

The iterative method should not win merely because:
This is an AI problem, therefore use an iterative AI training method.

It should win because:
Among the adequate available methods, this method provides the best justified computational choice.

The difference is procedural rather than ideological.

The Role of Direct and Non-Iterative Methods

The same neutrality applies to direct and non-iterative methods. A closed-form solution should not automatically be selected merely because it avoids iteration. Matrix factorization, decomposition, inversion, or pseudoinverse computation may itself be computationally expensive. A lookup procedure may require substantial memory. A rule-based system may require expensive construction and maintenance. An elimination procedure may fail when the information required for reliable elimination is unavailable.

APP therefore does not establish a universal hierarchy such as
RULES → LOOKUP → DIRECT ALGEBRA → ITERATION

No such ordering can be assumed universally.

Instead, APP treats different computational paradigms

as bidders whose adequacy and application-dependent computational prices must be evaluated. The framework is therefore method-neutral.

The Importance of the Ordinary User

The ordinary-citizen experiment highlights an important practical issue created by AI-generated software. A numerical analyst may know when QR, SVD, pseudoinverse, regularization, or another numerical procedure is appropriate. A machine-learning specialist may understand the implications of selecting Logistic Regression, SGD, Adam, Naive Bayes, or another method.

An ordinary user may know none of these. The user may simply have a problem.

As AI systems increasingly generate computational solutions for non-specialist users, requiring the user to identify the appropriate algorithms becomes increasingly unreasonable. The user should not need to know the name of the bidder.

The procurement procedure should find the bidders.

This represents an important distinction between traditional programming and AI-assisted programming. In traditional programming, the programmer explicitly selects much of the computational architecture. In AI-assisted programming, part of this architectural responsibility may be delegated to the AI system. With that delegation comes responsibility for computational selection.

From Post-Optimization to Pre-Selection Evaluation

A conventional workflow may be represented as

MODEL SELECTION
↓
TRAINING OR IMPLEMENTATION
↓
PERFORMANCE EVALUATION
↓
COMPUTATIONAL OPTIMIZATION

APP asks whether relevant computational considerations should enter earlier:

TASK AND REQUIREMENT DEFINITION
↓
CANDIDATE GENERATION
↓
ADEQUACY EVALUATION
↓
APPLICATION-DEPENDENT PRICE EVALUATION
↓
METHOD SELECTION
↓
EXECUTION OR TRAINING

The distinction is important.

APP does not require exhaustive optimization of every possible implementation before a method can be selected. Such a procedure could itself become computationally wasteful.

Instead, APP requires reasonable computational alternatives to be considered before a computational paradigm receives unjustified priority. This is not premature optimization. It is pre-selection evaluation.

The Cost of the Procurement Procedure

Algorithm procurement is not free. Generating candidate methods, evaluating their adequacy, estimating their computational prices, and comparing alternatives all require computational effort. This produces an important boundary condition for APP. The cost of searching for an improved computational method should not exceed the expected benefit produced by finding that method. Let

C_{proc} = computational price of the procurement procedure,

And

ΔC = expected computational saving or operational benefit obtained from improved method selection. Procurement is economically justified when, conceptually,

$\Delta C > C_{proc}$.

For a computational task executed only once, an extensive procurement procedure may therefore be unnecessary. For a task executed millions of times, even a small improvement in per-execution computation, latency, energy, or memory requirements may justify substantially greater procurement effort.

For a safety-critical application, the benefit may not be monetary at all. Improved latency, reliability, or mission performance may justify additional computational expenditure during method selection.

Thus, the procurement procedure itself must also satisfy the principle it imposes:
The cost of selection must be justified by the value of selection.

This prevents APP from becoming another unnecessarily expensive optimization procedure.

Limitations of the Present Study

The ordinary-citizen experiments presented here are exploratory. They do not establish that all AI systems behave similarly. They do not demonstrate that contemporary AI systems systematically select unnecessarily expensive methods. They do not establish that AI systems systematically prefer iterative optimization. They also do not establish that non-iterative methods are generally less expensive or more appropriate than iterative methods. Indeed, the first numerical experiment contradicted the initial suspicion that presenting a problem as an AI task would automatically produce an iterative solution.

The experiments instead identify a narrower behavior that can be tested systematically:
The computational method recommended by an AI system may depend on the task formulation and on the selection requirements explicitly included in the request.

A broader experimental study should therefore examine:
Multiple AI systems
Multiple Independent Sessions
Identical Blind Prompts
Different Prompt Formulations
Linear And Nonlinear Problems
Overdetermined Systems
Underdetermined Systems
Well-Conditioned And Ill-Conditioned Systems
Classification Problems
Repeated-Computation Problems
Small And Large Datasets
Real-Time And Latency-Constrained Problems
and Cases in which iterative optimization is expected to win.

Such experiments would allow APP and AI algorithm-selection behavior to be evaluated quantitatively rather than anecdotally. The present study therefore proposes a testable framework and experimental question rather than a universal conclusion about contemporary AI systems.

Computational Stewardship as an AI Responsibility

The preceding discussion raises a question extending beyond conventional algorithm selection.

If an AI system is entrusted with selecting a computational method and has access to knowledge of multiple applicable methods, what responsibility accompanies that authority?

The issue is different from asking whether the AI system knows an efficient algorithm.

The more important question is whether available computational knowledge is used appropriately during method selection.

An ordinary user may know the problem.

The AI system may know the available computational methods.

This creates an asymmetry of computational knowledge.

Requiring the user to discover the appropriate alternative and explicitly request it can therefore reverse the natural allocation of technical responsibility.

The experiments in this study illustrate this distinction. In some cases, alternatives did not have to be taught to the AI system. They entered consideration after the selection requirements were changed.

This leads to an important question:
If a reasonable alternative can be identified after additional questioning, should that alternative have entered the initial candidate set?
APP addresses this issue through computational stewardship.

Computational stewardship does not mean that the least

expensive algorithm must always be selected.

It does not mean that iterative optimization should be avoided.

Accuracy, robustness, numerical stability, scalability, reliability, latency, safety, and other requirements may justify a computationally more expensive method.

Computational stewardship instead requires reasonable alternatives to be considered and the selected method to be justified under the requirements of the application.

The responsibility can therefore be represented as
AVAILABLE COMPUTATIONAL KNOWLEDGE
↓
IDENTIFICATION OF REASONABLE ALTERNATIVES
↓
ADEQUACY EVALUATION
↓
APPLICATION-DEPENDENT COMPUTATIONAL PRICE
↓
JUSTIFIED METHOD SELECTION
The objective is not merely computational thrift.

It is responsible computational selection.

Accordingly:
The user should not have to identify an appropriate algorithm that the AI system can reasonably identify itself.

More generally:
When an AI system is given the authority to select the computational method, computational stewardship becomes part of that authority.

The algorithm is a bidder. The selection procedure is the procurement authority.

The User-First Argument

Computational stewardship does not require an individual user to minimize global computational consumption.

The user's immediate objective is to obtain an adequate solution to the problem.

APP begins from that objective.

If two methods satisfy the required conditions but one requires less computation, execution time, memory, hardware, energy, or operating expenditure without sacrificing required performance, the user has a direct engineering interest in knowing that alternative.

The argument therefore does not require an environmental or global-resource justification.
The first question is simpler:
Why should the user incur computational expenditure that is not required by the task?

Only after this individual consideration does scale become relevant.

If the same unnecessary computational expenditure occurs across many users and millions or billions of repeated executions, the aggregate effect may become a system-level engineering concern.

The order is therefore:
APP does not assume that existing AI systems are computationally inefficient.
USER REQUIREMENTS
↓
ADEQUATE SOLUTION
↓
BEST JUSTIFIED COMPUTATIONAL PRICE
↓
AGGREGATE SYSTEM BENEFIT

In many cases, deployed systems may already employ highly optimized algorithms, hardware, batching, caching, and other efficiency mechanisms invisible to the user.

Without measurement or sufficient system-level information, computational energy expenditure should not be asserted.

Computational cost must be measured, estimated under stated assumptions, or reported by the executing system.

Accordingly:
APP is not an accusation of inefficiency.

It is a formal requirement for justified computational

selection.

Algorithm procurement also does not require an exhaustive competition among all conceivable algorithms. The objective is not to conduct a computationally expensive search for the theoretically cheapest possible method. The objective is to prevent reasonable and evidently applicable alternatives from being excluded merely because another method is conventional or selected by default. The procurement procedure itself must remain method-neutral. Otherwise, algorithm procurement would merely replace one default preference with another.

The Behavior of Algorithm Selection in Artificial Intelligence

The exploratory experiments ultimately produced a broader research question than the one with which the study began.

The original concern was computational economy.

The first experiment then showed that an AI system could independently select a numerical least-squares procedure rather than automatically introducing gradient-based optimization. Later experiments showed that changing the problem structure, repeated-use conditions, numerical stability requirements, and computational-price criterion could change the methods brought into consideration. The broader research question is therefore:

How and why does an AI system select one computational method when several methods capable of performing the same task are available?

Several related questions follow:

Does the selected method change when accuracy, computational price, latency, energy, robustness, reliability, repeated use, or mission requirements are introduced explicitly?

What determines which algorithms enter the candidate set?

What causes an alternative method to be excluded?

Does the AI system already possess knowledge of an alternative that it did not initially consider?

Can algorithm-selection behavior be reproduced across systems, sessions, and prompt formulations?

The individual algorithms may already be known. The unresolved question is how and why an AI system brings particular algorithms into consideration, excludes others, and ultimately selects one computational procedure for a given task. This question extends APP beyond computational economy alone. It makes algorithm-selection behavior itself an object of experimental investigation.

Conclusions

This paper introduced the Algorithm Procurement Problem as a framework for justified selection among computational methods for artificial intelligence tasks. The central proposal is simple:

We do not impose an algorithm.

We impose an algorithm-selection procedure.

The procedure begins with the task and its requirements rather than with a preferred computational paradigm. Reasonable alternative methods are generated. Their adequacy is evaluated. Inadequate candidates are rejected. The application-dependent computational prices of the remaining candidates are compared. The task is then awarded to the best justified adequate computational alternative. The selected method may be iterative or non-iterative. It may involve GD, SGD, Adam, CG, direct algebra, pseudoinverse, regularization, Naive Bayes, lookup, rules, elimination, or another applicable method. APP does not determine the winner in advance. It determines how the winner should compete for selection.

The ordinary-citizen experiments provide preliminary evidence that algorithm-selection behavior may depend on the mathematical structure of the problem and on the requirements made explicit in the request. In particular, introducing computational price changed the candidate methods considered in one exploratory recognition experiment.

These observations do not establish a universal property of contemporary AI systems. They establish a testable research question. The future objective should therefore not merely be to make individual AI algorithms faster after they have been selected. The algorithm-selection process itself should become an object of computational

and experimental scrutiny. The final procurement principle can be stated as:

Do not select the algorithm first and ask its price later.

Include the price in the selection procedure.

Here, price is defined by the application and may include computation, latency, memory, energy, reliability, hardware, repeated use, mission requirements, or other relevant operational quantities.

The broader research question is:

How and why does an AI system select an algorithm?

The algorithm is a bidder. The selection procedure is the procurement authority. Artificial intelligence should not merely provide a computational solution. When it is entrusted with selecting the method, the basis for that computational selection should be justifiable.

Here, it is not being prescribed how AI should think. We are examining what enters the selection process when the AI is allowed to choose.

The central issue raised by APP is ultimately not the extent of the algorithmic knowledge available to an AI system. A sufficiently capable AI system may know gradient-based optimization, direct algebraic methods, statistical classifiers, lookup procedures, rule-based methods, elimination procedures, and many other computational alternatives. The important question is not simply what the system knows, but which of those alternatives it brings into consideration for a particular task, under what criteria, and why.

This distinction becomes increasingly important as AI systems become more capable. Greater capability gives an AI system a larger computational repertoire and greater freedom to interpret an underspecified user request. Consequently, algorithm selection may increasingly involve not only choosing among available methods, but also interpreting the requirements under which those methods should compete. Flexibility, extensibility, computational price, latency, reliability, numerical stability, energy, repeated use, or future adaptability may all be reasonable considerations, but their importance depends on the application and may not always have been explicitly specified by the user.

The question is therefore not whether an AI system should be prevented from making such judgments. Some interpretation is unavoidable, particularly when the user is not a specialist and provides only the objective of the task. Rather, the engineering question is whether the criteria materially influencing computational selection can be recognized and, when necessary, justified. This leads to a broader implication of the Algorithm Procurement Problem:

You may know every algorithm. The question is not what you know. The question is which alternatives you bring into consideration, under what criteria, and why.

As AI systems become increasingly capable of selecting computational methods on behalf of their users, this question becomes more important rather than less important. Greater algorithmic knowledge increases the number of possible bidders; greater autonomy in problem solving increases the importance of understanding the procurement procedure governing those bidders.

Accordingly:

The more capable the AI becomes at choosing algorithms, the more important it becomes to understand the criteria governing that choice.

APP therefore does not seek to restrict computational intelligence. It seeks to make better use of it. The objective is not to tell a capable AI system which algorithm must win, but to establish a selection process in which reasonable alternatives can compete and the resulting computational choice can be justified under the actual requirements of the task.

References

1. Schwartz R, Dodge J, Smith NA, Etzioni O (2020) Green AI. Commun ACM 63: 54-63.
2. Rice JR (1976) The algorithm selection problem. Adv Comput 15: 65-118.
3. Bischl B, Kerschke P, Kotthoff L, Marius Lindauer and Yuri Malitsky, et al. (2016) ASlib: A benchmark library for algorithm selection. Artif Intell 237: 41-58.
4. Thornton C, Hutter F, Hoos HH, Leyton-Brown

K (2013) Auto-WEKA: Combined selection and hyperparameter optimization of classification algorithm https://arxiv.org/abs/1208.3719.

5. He Y, Lin J, Liu Z, Wang H and Li LJ, et al. (2018) AMC: AutoML for model compression and hardware acceleration https://arxiv.org/abs/1802.03494.
6. Cai H, Gan C, Wang T, Zhang Z and Han S (2020) Once-for-All: Train one network and specialize it for efficient deployment https://arxiv.org/abs/1908.09791.
7. Kotthoff L, Thornton C, Hoos HH, Hutter F, Leyton-Brown K (2017) Auto-WEKA 2.0: Automatic model selection and hyperparameter optimization in WEKA. J Mach Learn Res 18: 1-5.

Copyright: ©2026 Huseyin Murat Cekirge. This is an open-access article distributed under the terms of the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original author and source are credited.