Electronics and Information Technologies, 2025, 32, 39–54
http://publications.lnu.edu.ua/collections/index.php/electronics/index

UDC: 004.04

SUSTAINABLE OPTIMIZATION OF CONSOLIDATED DATA PROCESSING ALGORITHMS BASED ON MACHINE LEARNING AND GENETIC ALGORITHMS

Vasyl Lyashkevych
Ivan Franko National University of Lviv,
50 Drahomanova St., 79005 Lviv, Ukraine

Lyashkevych, V.Y. (2025). Sustainable Optimization of Consolidated Data Processing Algorithms Based on Machine Learning and Genetic Algorithms. *Electronics and Information Technologies*, 32, 39–54. https://doi.org/10.30970/eli.32.3

ABSTRACT

Background. Automation of analytical report generation in industrial companies is gaining strategic importance due to the variety of document formats, increasing data volumes, and growing requirements for the rapid generation of multi-component analytical materials. Traditional ETL pipelines cannot cope with the complexity of modern information flows, especially when machine learning (ML), large language models (LLMs), and agent systems are integrated into the process. Due to the rapid progress of code generation and autonomous agent capability to perform complex analytical procedures, the task of automatically constructing reporting pipelines is becoming increasingly promising and scientifically sound.

Materials and Methods. An evolutionary model for constructing algorithms for processing consolidated reports based on genetic algorithms (GA) is proposed. For report generation, the algorithm defines a pipeline for constructing a visual component. The population grows as a tensor, enabling parallel evolution of a set of independent workflows. The operations are classified into four groups: ETL, ML, LLM, and VIS. The fitness function evaluates the constant length of the pipeline, the coverage of key types of operations, and their structural consistency.

Results and Discussion. Experimental results have shown that GAs rapidly evolve from random NOP-dominated structures to stable, logically consistent, and functional pipelines with a duration of 10-14 operations. The best chromosomes formed three full-fledged visual components: a predictive regression model, semantic clustering represented by embeddings, and a categorical diagram. This evolutionary pattern confirms that combined pipelines can be built automatically and adaptively, and the increase in the complexity of operations in the “meaning” space, which is the vector space of embedding, along with the development of code generation and agent architectures.

Conclusion. The proposed model demonstrates an effective mechanism for automated synthesis of multi-visual reports based on evolutionary pipelines. The method's prospects grow with the development of AI agent systems and the increase in the number of operations in the content space, which paves the way for a complete system of autonomous analytical reporting of a new generation.

Keywords: consolidated data, sustainable optimization, machine learning, genetic algorithms, LLM, data analytics

INTRODUCTION

In modern information ecosystems, where industrial companies operate with large volumes of heterogeneous data, there is a growing need not only for the automation of analytical processes, but also for sustainable approaches to optimizing consolidated data (CD) processing. The sustainability of algorithms refers to their ability to remain effective as formats change, workload increases, data sources expand, and business requirements evolve [1-2]. CD processing includes extraction, transformation, semantic structuring, and information integration that requires adaptive intelligent mechanisms capable of maintaining long-term stability, minimizing costs, and ensuring scalability in dynamic environments [3].

In this context, a combination of ML methods and GAs proves to be effective. In AutoML systems, GAs are successfully used to optimise hyperparameters, select models, create and reduce pipelines, and dynamically adjust their structure [4-7]. Due to the ability of evolutionary algorithms to search for optimal configurations in large solution spaces, their integration into industrial report processing tasks looks natural and promising. They will not be able to build evolutionary-adaptive pipelines that maintain system stability, adapt to the emergence of new formats, and minimise operational costs [8].

In conditions of a global economy, industrial and commercial enterprises interact with a wide network of suppliers and customers. It generates complex and highly variable data flows. Data comes in hundreds of disparate formats such as spreadsheets, PDFs, presentations, graphical reports, scanned forms, and hybrid structures. Usually, they vary significantly in terms of language, structure, and content. Automating such processing becomes a critical task to ensure consistency and quality of corporate analytics. Current research demonstrates the effectiveness of using LLMs, vector and graph databases [9-11], multi-agent systems [12-13], ML, deep learning (DL), and reinforcement learning (RL) methods [14-16], as well as evolutionary and GAs [6-7, 17] for optimizing ETL pipelines and analytical processes.

A separate research direction involves combining LLM with vector databases to enhance the quality of semantic search, classification, and context analysis [9]. Other works emphasise the potential of graph databases in combination with multi-agent architectures for implementing multi-step logical inference [13]. Research on optimising ETL processes using ML/RL suggests the possibility of dynamically adapting pipelines, detecting anomalies, and automatically selecting transformations [16, 18].

Considerable scientific interest is also associated with the role of LLMs in performing analytical operations in the “meaning space.” The work [19] demonstrated that LLMs are capable of performing interpretive, logical, statistical, and arithmetic operations without the use of hard-coded algorithms, operating on data in the semantic (“meaning”) space. This opens opportunities for building flexible, dynamic exploratory data analysis (EDA) processes without predefined pipelines, which is especially important in conditions of a wide variety of report formats and changing requirements.

Despite these advances, industrial reporting systems face several unresolved challenges:

1. A growing variety of formats, over a hundred or more;
2. The need to simultaneously extract transactional, contextual, and semantic data;
3. High demands on accuracy, consistency, and robustness of etl processes;
4. The need to generate new reports with different types of visualisations;
5. The need for adaptive pipeline evolution in response to the dynamics of formats and business requirements demanded in the market.

A particular problem is the high cost and instability of real experiments in an industrial environment. Testing different combinations of ETL operations, ML/LLM modules, and pipeline structures requires significant resources and time, and can lead to quality losses

or failures in productive systems. Therefore, there is a need for preliminary modelling using a simulation environment that allows for evaluating the effectiveness of various configurations without their physical implementation.

In this context, GAs are a natural tool for modelling, optimizing, and adaptively evolving report processing pipelines. Their strength lies in the fact that:

1. ETL, ML, and LLM operations can be represented as genes, and the pipeline as a chromosome.
2. GA allows for minimizing the number of operations while maintaining or improving the quality of the results.
3. GAs provide the search for the optimal sequence of operations according to contextual variables and requirements such as accuracy, time, resources, quality of the final report, required visuals, and others.
4. GAs allow for simulation of the system before its actual implementation and deployment, reducing risks and costs.
5. GAs naturally adapt to the emergence of new formats and new types of operations.

As mentioned, GAs are successfully used in AutoML systems, so there is a reasonable scientific expectation that, in the context of the current task, with the extraction of transactional, contextual, and semantic data from heterogeneous reports, the application of GAs will be extremely effective. The integration of LLM approaches in the semantic space [19], multi-level storage systems which include SQL, VectorDB, and GraphDB, multi-agent architectures, ML/DL/RL modules, and GAs forms the foundation of a new paradigm - an evolutionary-adaptive system for sustainable optimization of CD processing algorithms which can automate processing of industrial reports.

Recent advances in code generation using LLMs have significantly expanded the capabilities of automated data-processing systems, enabling dynamic construction of ETL procedures, semantic extraction modules, and visualization logic directly from natural-language specifications. State-of-the-art models such as Codex, CodeT5, GPT-4 and Llama-3 have demonstrated near-human performance in generating reusable and semantically consistent code components for data analytics workflows [20-22]. Scientific studies show that LLM-driven code generation not only accelerates pipeline development but also improves modularity, reproducibility, and correctness through learned structural patterns and contextual reasoning [23]. When applied to automated reporting, code-generating models create flexible and adaptive procedures for parsing, embedding, aggregating, storing, and visualizing data, enabling rapid reconfiguration of workflows in response to new report formats or analytical requirements. Thus, now it no longer makes sense to program all technological operations in advance and after serving them.

Simultaneously, agentic AI systems have emerged as a powerful paradigm for orchestrating multi-step analytical processes. Research demonstrates that multi-agent architectures, comprising planning agents, reasoning agents, tool-using agents, and code-executing agents, achieve superior performance in complex, multi-stage tasks such as document analysis, information synthesis, and dynamic pipeline construction [24-25]. Agentic systems benefit from autonomous task decomposition and iterative refinement, allowing agents to negotiate, supervise, and correct one another, thereby reducing error propagation and increasing robustness. When applied to automated report generation, agentic AI can coordinate the entire lifecycle: ingesting documents, selecting transformations, generating executable code, validating results, and producing narrative and visual outputs.

Before constructing a fully autonomous agentic pipeline, it is essential to first identify the optimal algorithmic structure for the report-building workflow. Agentic systems rely on a predefined space of tools and capabilities. If this space is suboptimal, redundant, or contains low-quality operations, the agents will form inefficient or even failed workflows.

Scientific evidence shows that agent-based systems are highly sensitive to the structure of available operations, often suffering from combinatorial explosion and suboptimal planning when the action space is not pre-optimized [24]. Therefore, determining the optimal algorithm, through GAs, evolutionary search, or analytical design, ensures that the agentic system operates within a validated, minimal, high-quality set of operations, maximizing performance and sustainability. Only after this optimization step, agentic AI can reliably construct scalable, interpretable, and efficient multi-visual report generation pipelines.

MATERIALS AND METHODS

Modern industrial companies support reports from numerous suppliers and customers in hundreds of disparate formats that differ in structure, data types, language, context, and semantic relationships. Those reports contain different types of CD, such as transactional, contextual, and semantic. This needs to be properly extracted, transformed, and integrated into the appropriate SQL, VectorDB, and GraphDB repositories, respectively. The growth in the number of formats, the dynamic nature of changing business requirements, and the need for scalability create a demand for continuous, sustainable optimization of CD processing processes.

In real industrial conditions, testing all possible combinations of ETL, ML, and LLM operations is extremely expensive, risky, and unsustainable: it requires significant time and computing resources, can disrupt productive processes, and lead to data quality losses. Therefore, the primary task is to create a simulation model that allows for virtually exploring and optimising data processing pipelines before their physical deployment. The most suitable tool for such optimization is GAs. Thanks to the evolutionary operators of selection, crossover, and mutation, GAs can effectively explore a large space of possible processing configurations and find stable and minimally complex pipeline variants.

Formalization of the problem

The data processing and analytics pipeline is shown in **Fig. 1**. Users can provide input reports in various formats, such as PDF, XLSX, databases, and APIs containing raw data to the pipeline. An algorithmic workflow processes the data through multiple stages: O_1, O_i, O_j and O_l, extracting insights and transforming it. The system integrates data from multiple storage systems, including SQL databases, VectorDBs, and GraphDBs, ensuring compatibility with various data formats.

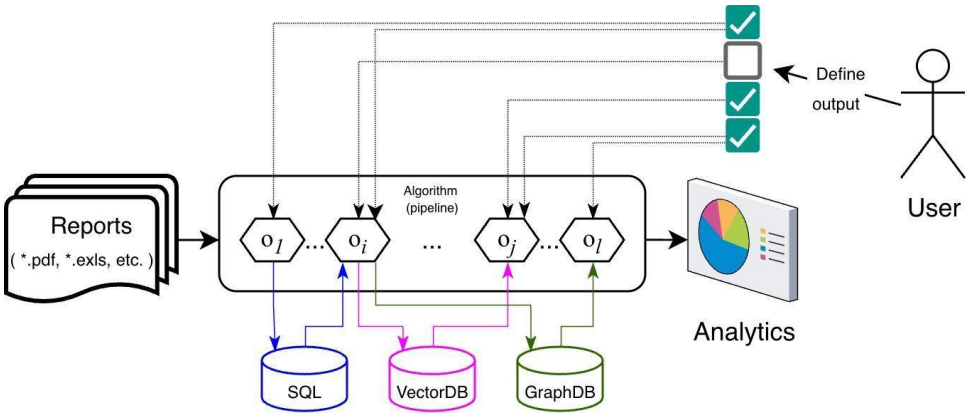

Fig. 1. Report generation approach.

A user defines the desired outputs as visual components, which determine the analysis scope, metrics, and visualizations in the report. Based on these preferences, the

processed data is transformed into tailored analytics outputs, such as visual charts or custom reports.

This architecture supports flexible processing, allowing dynamic user-driven customization of outputs and efficient multi-database integration. It is particularly useful for applications where decision-making relies on consolidated, processed insights from diverse and complex datasets and for automation based on an Agentic AI or multi-agent approaches.

Let the industrial report processing system be constructed as an evolutionary-adaptive algorithm A, consisting of an ordered set of operations:

$$A = \{o_1, o_2, \ldots, o_l\}, \; o_i \in O, \tag{1}$$

where O is the universal set of all possible operations (ETL, ML, LLM, Visualization).

Each operation is defined as:

$$o_i = (t_i, \theta_i, D_i), \tag{2}$$

where: t_i is type of operations (ETL, ML, LLM, Visualization); θ_i – parameters of operations; $D_i = \subseteq O \times O$ – a set of dependencies (pre- and post-conditions).

There are three main types of dependencies in CD processing: structural, semantic, and functional. In case of structural dependencies, an operation o_j may only be executed after the other operation o_i, for example $o_i \preccurlyeq o_j$. Formally:

$$\forall o_j \in A: Pre(o_j) \subseteq A. \tag{3}$$

The operation o_j requires semantic entities produced earlier:

$$Entities(o_j) \subseteq \cup_{i<j} Output(o_i), \tag{4}$$

Example: “LLM summarization” depends on “context normalization”, which depends on “Entity extraction”.

Operation o_j may require a specific type of input:

$$TypeIn(o_j) = TypeOut(o_i). \tag{5}$$

Example: “ML classification” requires embeddings produced by a “vectorization operation”.

Therefore, a CD processing algorithm A is valid when the conditions (3), (4), and (5) are met simultaneously. Thus, GA evolves not arbitrary sequences but structurally valid algorithms.

The context of a document $c \in C$ constrains which operations may be applied:

$$O_c = \{o \in O \mid Applicable(o, c) = 1\}. \tag{6}$$

Example of context-dependent operations:

- if a document is in Chinese → translation “+” multilingual normalization required;
- If PDF → OCR and structural extraction are required;
- If it contains graphical tables → table reconstruction is needed.

Thus: $A_c \subseteq O_c$.

Data transformation through the algorithm A is defined as:

$$X_0 = r \in R; \quad X_{i+1} = o_i(X_i); \quad X_l = A(r). \tag{7}$$

Required outputs: transactional data $T(r)$, contextual embeddings $V(r)$, and semantic graph $G(r)$. Algorithm correctness condition $T(r), V(r), G(r) \neq \emptyset$:.

The optimization objective of the fitness function with dependency penalties is:

$$F(A) = \beta_1 E(A) + \beta_2 L(A) - \beta_3 Q(A) - \beta_4 S(A) + \beta_5 D(A), \tag{8}$$

where: $E(A)$ is number of operations, $L(A)$ – latency/compute cost, $Q(A)$ – extraction quality such as accuracy, recall, and consistency, $S(A)$ – sustainability/adaptiveness, $D(A)$ – dependency violation penalty. Penalty:

$$D(A) = \lambda \cdot |\{o_i \in A \mid not\ Valid(o_i)\}|, \tag{9}$$

If the algorithm satisfies all dependencies, it will be 1, otherwise 0. Therefore, algorithm validity, using (3), (4), and (5), is being calculated:

$$Valid(A) = \wedge_{j=1}^{l}\left[Pre(o_j) \subseteq \{o_1, \ldots, o_{j-1}\}\right] \wedge \left[TypeIn(o_j) = TypeOut(o_{j-1})\right]. \tag{10}$$

Finally, the optimization objective:

$$A* = arg\ min_{P \in P_{valid}} F(A). \tag{11}$$

The goal is to find and evolutionarily optimize a data-processing algorithm, which emulates a report processing pipeline, that satisfies all structural, functional, and semantic dependencies between operations while minimising the number of operations and computational cost, maximizing the extraction quality of transactional, contextual, and semantic data, and maintaining adaptability to new report formats.

Sets of operations for experiments

Defining sets of ETL, ML, LLM, and visualization operations is essential for building a formal, managed, and optimized report processing process. When a system receives heterogeneous reports, it must know what transformations to perform: what data to parse, what embeddings to create, how to aggregate, and where to store the information. A clear separation of operations allows building scalable pipelines that support three data types: transactional (SQL), contextual (VectorDB), and semantic (GraphDB). It also provides automatic selection of ML, LLM, and visualization components and optimization of their sequence using GAs. Examples of operations are shown in **Table 1**.

The global operation space can be described as:

$$O = O_{ETL} \cup O_{ML} \cup O_{LLM} \cup O_{VIS}, \tag{12}$$

where: O_{ETL} are operations for loading, converting, merging and storing data; O_{ML} – operations for statistical and machine learning; O_{LLM} – semantic, linguistic and reasoning operations; O_{VIS} – operations for building reports and visualizations.

Combined operation space is a unified space of all possible operations that a system can use for automated report processing. It combines four large classes: ETL, ML, LLM, and Visualization. Each operation performs a specific function such as parsing, cleansing, aggregation, classification, semantic extraction, text generation, or visualization of results.

Table 1. **Sets of used operations in experiments**

Operation	Functionality	Samples
O_{ETL}^{ing}	Input, upload, and ingestion	UploadFile, DownloadFromAPI, IngestEmail, FetchFTP
O_{ETL}^{parse}	Parsing and extraction	PDFParse, OCR, TableDetection, ExcelParse, CSVParse, ImageToTable, HTMLParse
O_{ETL}^{norm}	Data normalization	CleanText, NormalizeNumbers, Deduplicate, FixEncoding, DateNormalization, CurrencyNormalization
O_{ETL}^{agg}	Data aggregation and fusion	JoinRecords, MergeSources, GroupBy, Summarize, Pivot, Unpivot
O_{ETL}^{store}	Data storing	StoreSQL(T), StoreVectorDB(V), StoreGraphDB(G)
O_{ETL}^{read}	Data reading	ReadSQL, ReadVector, ReadGraph
O_{ML}^{clas}	ML classification	DocumentTypeClassifier, AnomalyClassifier, SupplierCategoryClassifier
O_{ML}^{reg}	ML regression	PredictMissingValues, ForecastQuantities, CostRegression, DeliveryTimeRegression
O_{ML}^{clust}	ML clustering	ClusterReports, ClusterSuppliers, ClusterContentTopics
O_{ML}^{emb}	Embedding-based ML	ComputeEmbeddings, SemanticSimilarity, NNRetrieval
O_{LLM}^{reason}	Semantic understanding and reasoning	SemanticParsing, LogicalInference, MultiHopReasoning, ConstraintValidation
O_{LLM}^{gen}	Summarization and generation	ShortSummary, DetailedSummary, ExecutiveReport, NarrativeGeneration
$O_{LLM}^{extract}$	Information extraction	EntityExtraction, RelationExtraction, AttributeExtraction, SchemaInduction
O_{LLM}^{trans}	Transformation	Translate, NormalizeTerms, Rewrite, CleanNoise, ContextualRewriting
O_{LLM}^{etl}	LLM for ETL	LLM-TableReconstruction, LLM-DataValidation, LLM-SQLQueryGeneration
O_{VIS}^{basic}	Basic charts	BarChart, LineChart, PieChart, Histogram, ScatterPlot
O_{VIS}^{adv}	Advanced analytics visuals	Heatmap, Sankey, GeoMap, TimeSeriesForecastPlot, CorrelationMatrix
O_{VIS}^{narr}	Narrative visualization	ChartExplanation, NarrativeAnalytics, LLM-GeneratedReport
NOP	No Operation	NOP

Combining these sets into a single space O allows:

- to consider any operation as a gene in the pipeline chromosome;
- to build flexible, adaptive, and diverse pipelines;
- allow the GA to freely combine and optimize different types of operations;
- provide support for all data types: transactional (SQL), contextual (VectorDB), and semantic (GraphDB).

Thus, the combined operation space is a foundation that utilizes all the transformation capabilities that the system can perform on data. Thus, for example, the combined operation space can be defined:

$$O = O_{ETL}^{ing} \cup O_{ETL}^{parse} \cup O_{ETL}^{norm} \cup O_{ETL}^{agg} \cup O_{ETL}^{store} \cup O_{ML} \cup O_{LLM} \cup O_{VIS}, \quad (13)$$

Every element O can be gene in a chromosome, forming an executable pipeline.

General model of operation

Each operation should be formalized within a unified operational specification suitable for LLM, ML, ETL, and Visualization classes. Thus, an operation is described as:

$$o = (id, type, X_{in}, X_{out}, \theta, C, Dep), \quad (14)$$

where: id is unique transaction identifier; $type$ – class of ETL, ML, LLM or VIS; X_{in} – type of input data; X_{out} – type of output data; θ – operation parameters; C – contextual execution conditions and Dep – depending on other operations.

Prepare GA for experiments

Encoding operations and chromosome structure. Each operation from the combined operation space (ETL, ML, LLM, VIS) is encoded as a gene, represented by a unique integer or symbolic token:

$$o_i \rightarrow g_i \in \Sigma, \quad (15)$$

where Σ is the operation alphabet.

In the classical formulation of evolutionary pipelines, a chromosome is represented as a single linear sequence of operations:

$$P = [g_1, g_2, \dots, g_L], \quad (16)$$

where each gene corresponds to a transformation step in a pipeline.

Because different pipelines may require different numbers of operations, we use a fixed-length chromosome. A fixed length L guarantees comparability among individuals and stable crossover or mutation behaviour. To allow variable-length logic within a fixed-length chromosome, unused positions are filled with a NOP gene:

$$NOP(X) = X. \quad (17)$$

The NOP operator allows the GA to simulate shorter pipelines within a fixed-length representation, supports incremental growth of solutions, and prevents destructive crossover.

However, this representation encodes only one computational workflow, suitable for generating a single analytical output (one visualization or one analytical component). In the context of automated report generation, a user typically requires multiple analytical outputs simultaneously, such as n different charts, KPIs, summaries, embeddings, graphs or others. Each of these outputs must be built by a distinct computational pipeline, possibly sharing some operations but often diverging after early stages. Therefore, a single linear chromosome is insufficient.

Typical document-processing pipelines include from 20 to 50 operations, depending on operational diversity. A single pipeline builds only one analytical output, but real reports

require multiple independent visuals. In this case, the chromosome must be represented as a matrix of size $L \times n$. Consequently, the GA population (**Fig. 2**) becomes a tensor $M \times L \times n$, enabling simultaneous evolution of multiple pipelines under shared constraints:

$$P \in R^{L \times n}, \tag{18}$$

where: L is max pipeline length (50 operations per one visual or analytical component), n – number of requested visuals or analytical components, each column $P_{i,j}$ encodes one full algorithmic pipeline for visual j.

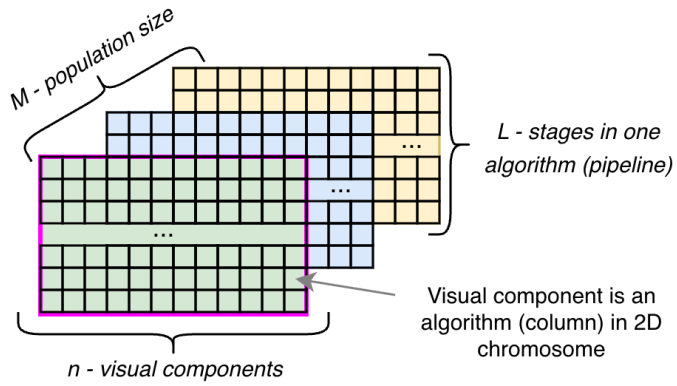

Fig. 2. Structure of a chromosome.

With matrix chromosomes, the population is no longer a 2-D matrix (population size × genome length). It becomes a 3-dimensional tensor:

$$P \in R^{M \times L \times n}, \tag{19}$$

where: M is population size, L – pipeline length per output, n – number of independent visuals.

The proposed chromosome's structure considers:

- Multi-objective report generation. A report with n visuals is not a single-objective optimization problem but a multi-pipeline multi-objective system. Each visual requires a different data-processing workflow.
- Preservation of algorithm (pipeline) independence. Encoding all visuals in a single vector would mix operations across visuals and destroy semantic locality. Matrix representation preserves the independence of pipelines.
- Parallel evolution in a shared context. Early operations such as parsing, OCR, and aggregations may be shared, while later stages such as special aggregations, ML, LLM, or visualization diverge. Matrix form allows partial sharing naturally.
- Tensor representation enables proper genetic operators. Mutation and crossover can be applied, such as column-wise (per visual), row-wise (per pipeline depth), and block-wise (submatrix crossover).
- Scalable and sustainable optimization. Industrial reports frequently contain 10-20 visuals, avoiding redundant computation and ensuring consistency.

The initial population is created using heuristic seeding. We assume that a subset of 10-20% of the population is initialized using heuristically valid partial pipelines:

- parsing → normalization → embedding → ML/LLM → storage;

- OCR → table reconstruction → aggregation → storage;
- parsing → LLM extraction → graph building.

This accelerates convergence.

Some individuals are initialized with many NOP genes and a small number of operations inserted in valid positions. This encourages smooth evolutionary growth from simple to complex solutions.

RESULTS AND DISCUSSION

The proposed GA was evaluated through controlled simulations designed to model the automated construction of multi-visual analytical reports. Due to the limited quantity of the prepared operations, we cannot allow the use of a large population. Thus, the experiment used a population of 40 matrix-encoded chromosomes, each representing a set of three parallel pipelines (one per visual) with a maximum depth $L = 30$ of operations. Thus, each chromosome formed a tensor-shaped individual of size 30×3, while the entire population spanned a tensor of size $40 \times 30 \times 3$. Such representation enabled the simultaneous evolution of multiple workflows required for multi-visual reporting, which cannot be encoded in a classical one-dimensional chromosome.

Fitness function evaluation

The GA was executed for 50 generations. Fitness was calculated per visual using a function that rewarded: sustainable pipeline length (~10 operations), presence of ETL, ML, LLM, and VIS categories, and overall completeness of the workflow.

For each individual, the fitness is computed per visual (per column) and then summed across visuals:

$$F(P) = v = \sum_{v=1}^{N_{vis}} f(P_v). \tag{20}$$

Example of calculations for one visual component:

$$f = -|\, effective_{length} - 10\,| + 3 \cdot has_{ETL} + \\ +2 \cdot has_{ML} + 2 \cdot has\_LLM + 4 \cdot has\_VIS.$$

As is seen from the provided example, we expect approximately 10 operations in the pipeline as an effective length. GA optimization dynamics (**Fig. 3**) show rapid improvement during the first 10 generations, followed by gradual stabilization.

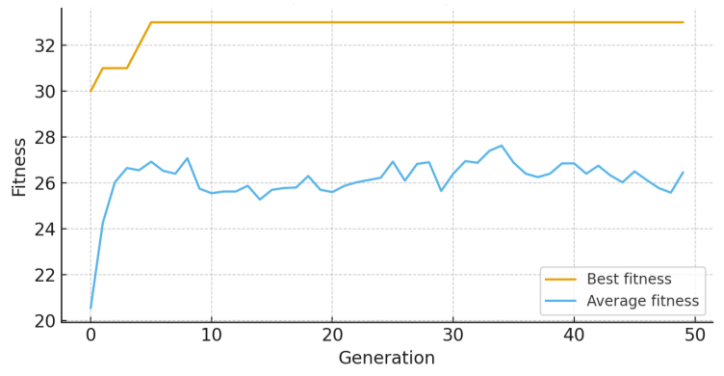

Fig. 3. Fitness function estimation.

The best fitness converged at 33, while the population mean stabilised around 26. This indicates that the GA successfully moved from sparse, primarily NOP-heavy pipelines toward meaningful multi-stage workflows, combining ETL, ML, LLM, and VIS operations.

Pipeline length distribution

Fig. 4 shows the distribution of effective pipeline lengths for the final generation. The population converged around 10-14 operations per pipeline, which aligns with the sustainability objective encoded into the fitness function.

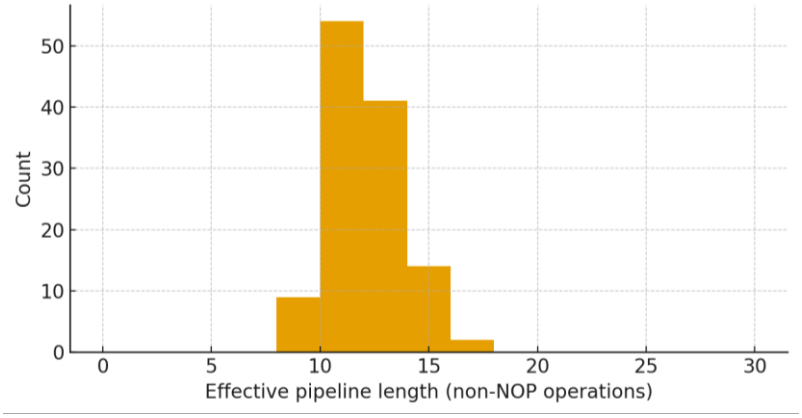

Fig. 4. Pipeline length distribution.

Pipelines shorter than 6 steps lacked analytic capability, while pipelines longer than 15 steps accumulated penalties due to inefficiency. The GA thus demonstrated an emergent preference for compact, interpretable, resource-efficient pipelines.

Pipeline decoding

The best-performing chromosome consisted of a 30×3 matrix, in which three distinct pipelines were evolved. Each pipeline included a mixture of:

- ETL steps: parsing, normalization, and table extraction;
- ML operations: classification, embedding computation, regression, or clustering;
- LLM methods: semantic parsing, summarization, and textual reconstruction;
- Visualization components for final chart generation.

Decoding the matrix revealed three semantically coherent workflows:

- visual 1: ETL → ML → LLM → VIS, suitable for KPIs and trend analysis;
- visual 2: ETL → ML → LLM → VIS, producing semantic cluster views;
- visual 3: ETL → LLM → VIS, producing narrative-style or categorical summaries.

The existence of all four operation classes in the final chromosome confirms that the GA learned the implicit structure of real-world analytical reporting pipelines, even though no explicit constraints, beyond validity and type matching, forced such ordering. To validate that the evolved pipelines correspond to meaningful outputs, three synthetic visuals were generated:

- The regression forecast chart (**Fig. 5**) represents time-series prediction, illustrating the ML segment of the pipeline.
- Semantic embedding clusters (**Fig. 6**) represent LLM with ML synergy for contextual analytics.
- The categorical summary chart (**Fig. 7**) represents ETL aggregation followed by basic visualization.

These visuals emulate typical report components found in automated dashboards: trend analysis, semantic grouping, and KPI summaries.

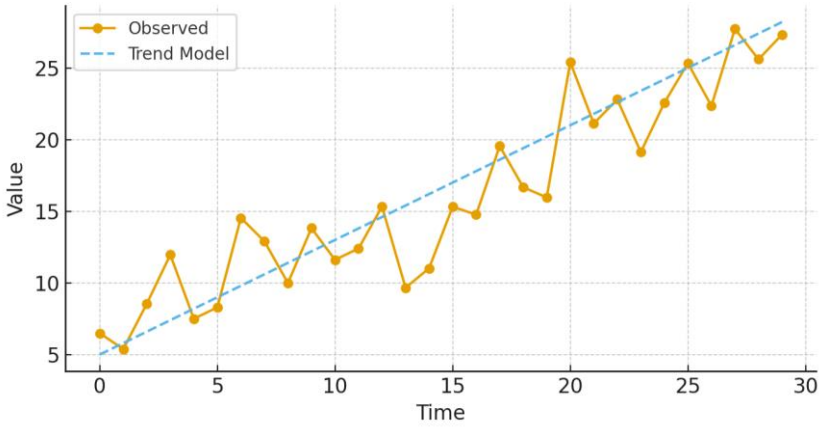

Fig. 5. Regression forecast chart.

The ability of the evolved pipelines to generate full visuals demonstrates that the GA was able to construct functional workflows, not just syntactic sequences of operations.

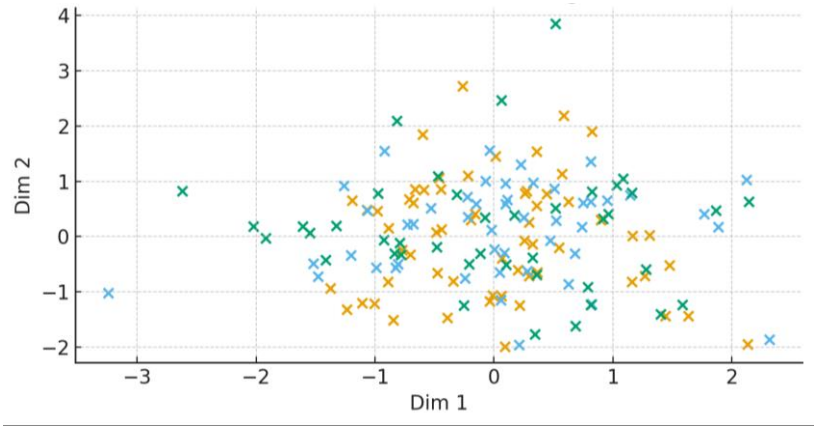

Fig. 6. Semantic embedding clusters.

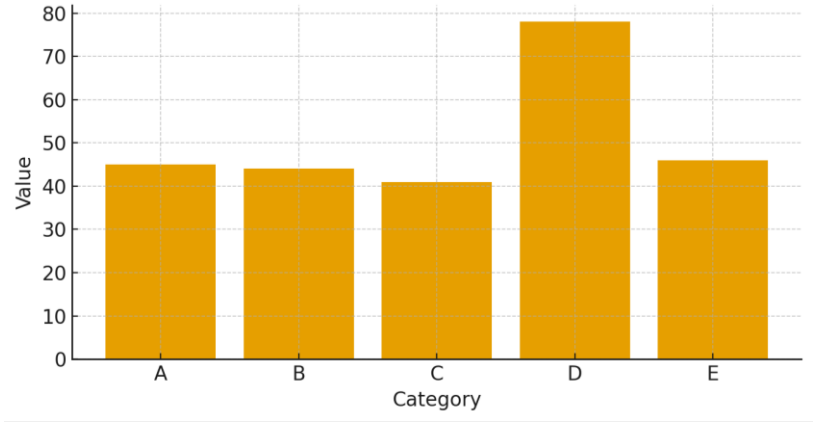

Fig. 7. Categorical summary chart.

The experimental results confirm several critical findings:

- Matrix chromosome encoding works. Encoding the chromosome as a matrix (pipeline-per-visual) supports parallel workflow evolution and allows GA to optimize complete multi-visual reports.
- GA can learn sustainable algorithms (pipelines). The convergence around 10-14 operations demonstrates that the fitness function effectively guides the algorithm toward sustainable ETL, ML, LLM, and VIS chains.
- Automatic multimodal report construction is feasible. The final pipelines successfully generated three distinct and meaningful visuals.
- Emergency structure matches human workflow design. Despite no explicit manual design, the GA consistently evolved a logical ordering: ETL first (ingestion, parsing, cleaning), ML/LLM next (interpretation, semantic enhancement) and visualization last (final chart generation).
- Population-level diversity was maintained. Although fitness converged, the distribution of pipeline structures remained diverse, indicating that mutation and crossover operators preserved exploration.

These results support the viability of using a tensor-based GA for industrial report automation, especially in heterogeneous data environments requiring simultaneous multimodal analytics.

CONCLUSION

This research demonstrates that evolutionary optimization, when combined with modern ML, LLMs, and agentic AI principles, provides an effective methodological foundation for automated report generation in heterogeneous industrial environments. The developed tensor-based GA model successfully evolves multi-stage, multi-visual analytic pipelines by representing each report as a matrix of parallel workflows and optimizing them jointly. Experimental results confirm that the proposed framework converges toward sustainable, interpretable, and structurally valid pipelines, balancing ETL, ML, LLM, and VIS operations into coherent analytic sequences.

The study also reveals that as the operational space expands, through the growth of meaning-space operations, improved code-generation capabilities, and the emergence of agentic AI systems, the effectiveness and scalability of evolutionary pipeline synthesis increase substantially. These technological developments amplify the adaptability and autonomy of the system, enabling dynamic reconfiguration of analytic workflows in response to new formats, data modalities, and reporting objectives. Consequently, evolutionary optimization becomes a foundational step for future agentic systems. It establishes an optimized, validated and semantically coherent action space from which AI agents can reliably plan, coordinate and execute reporting tasks. The findings thus position evolutionary algorithm-driven pipeline synthesis as a crucial enabling technology for next-generation autonomous analytic ecosystems.

The research introduces a novel encoding of analytic pipelines as a matrix L × N rather than a linear sequence, enabling simultaneous evolution of multiple report visuals within a single chromosome. This produces richer, more scalable, and structurally coherent solutions compared to classical GA pipeline representations.

The proposed framework integrates heterogeneous operation types into a single evolutionary unified multimodal operation space “ETL-ML-LLM-VIS”. This unification is unprecedented in existing AutoML or AutoETL systems. The experiments prove that complex reporting pipelines can emerge autonomously from evolutionary pressure, without manually encoded domain rules, which is an important step toward self-assembling analytics systems.

A key contribution of this work is the formalization of LLM-based semantic operations, such as summarization, multi-hop reasoning, entity and relation extraction, and contextual

normalization, as evolvable genetic components within the pipeline. Treating meaning-space transformations as genes places them on equal theoretical footing with classical feature engineering, ML, and ETL operations, thereby extending the domain of evolutionary computation from structural data manipulation into the realm of semantic cognition.

The study formulates a new design principle: before constructing agentic AI systems capable of autonomous report generation, the underlying operation space must be optimized through evolutionary search. This ensures that the action space available to agents is minimal, non-redundant, sustainable, and functionally validated.

ACKNOWLEDGMENTS AND FUNDING SOURCES

The author received no financial support for the research, writing and publication of this article.

COMPLIANCE WITH ETHICAL STANDARDS

The author declares that the research was conducted in the absence of any conflict of interest.

AUTHOR CONTRIBUTIONS

The author has read and agreed to the published version of the manuscript.

REFERENCES

[1] Fan, J., Han, F., & Liu, H. (2014). Challenges of Big Data analysis. National Science Review, 1(2), 293–314. https://doi.org/10.1093/nsr/nwt032

[2] Fernandes, A. A. A., Koehler, M., Konstantinou, N., et al. (2023). Data preparation: A technological perspective and review. SN Computer Science, 4(425). https://doi.org/10.1007/s42979-023-01828-8

[3] Ahlawat, P., Borgman, J., Eden, S., Huels, S., Iandiorio, J., Kumar, A., & Zakahi, P. (2023). A new architecture to manage data costs and complexity. BCG. https://on.bcg.com/3HOP7vQ

[4] Kwon, N., Comuzzi, M. (2023). Genetic Algorithms for AutoML in Process Predictive Monitoring. In: Montali, M., Senderovich, A., Weidlich, M. (eds) Process Mining Workshops. ICPM 2022. Lecture Notes in Business Information Processing, vol 468. Springer, Cham. https://doi.org/10.1007/978-3-031-27815-0_18

[5] Shi, K., Saad, S. (2023). Automated feature engineering for AutoML using genetic algorithms. In Proceedings of the 15th International Joint Conference on Computational Intelligence (pp. 450-459). SCITEPRESS. https://www.scitepress.org/Papers/2023/120904/120904.pdf

[6] Hernandez, J., Saini, A., Ghosh, A., Moore, J. (2025). The tree-based pipeline optimization tool: Tackling biomedical research problems with genetic programming and automated machine learning. Patterns. 6. 101314. 10.1016/j.patter.2025.101314. https://doi.org/10.1016/j.patter.2025.101314

[7] Jiao, J., Yuan, J. (2025). GA-PRE: A Genetic Algorithm-Based Automatic Data Preprocessing Algorithm. In Proceedings of the Genetic and Evolutionary Computation Conference (GECCO '25). Association for Computing Machinery, New York, NY, USA, 1371–1378. https://doi.org/10.1145/3712256.3726312

[8] Polonskaia, I. S., Nikitin, N. O., Revin, I., Vychuzhanin, P., & Kalyuzhnaya, A. V. (2021, June). Multi-objective evolutionary design of composite data-driven models. In 2021 IEEE Congress on Evolutionary Computation (CEC) (pp. 926-933). IEEE. https://doi.org/10.1109/CEC45853.2021.9504773

[9] Jing, Z., Su, Y., Han, Y., Yuan, B., Liu, C., Xu, H., & Chen, K. (2024). When Large Language Models Meet Vector Databases: A Survey. arXiv. https://arxiv.org/abs/2402.01763

[10] Sequeda, J., Allemang, D., & Jacob, B. (2025). Knowledge graphs as a source of trust for LLM-powered enterprise question answering. Journal of Web Semantics, 85, 100858. https://doi.org/10.1016/j.websem.2024.100858
[11] Instaclustr. (2024). *Vector Databases and LLMs: Better Together*. https://www.instaclustr.com/education/open-source-ai/vector-databases-and-llms-better-together/
[12] DeepFA AI. (2025). Multi-Agent Systems in Artificial Intelligence. https://deepfa.ir/en/blog/multi-agent-systems-artificial-intelligence
[13] Ramachandran, A. (2025). *Revolutionizing Knowledge Graphs with Multi-Agent Systems: AI-Powered Construction, Enrichment, and Applications*. ResearchGate
[14] Mehta, V., Batra, N., Poonam, Goyal, S., Kaur, A., Dudekula, K. V., & Victor, G. J. (2024). Machine Learning Based Exploratory Data Analysis and Diagnosis of Chronic Kidney Disease (CKD). https://doi.org/10.4108/eetpht.10.5512
[15] Da Poian, V., Theiling, B., Clough, L., McKinney, B., Major, J., Chen, J., & Hörst, S. (2023). Exploratory data analysis (EDA) machine learning approaches for ocean world analogue-mass spectrometry. https://doi.org/10.3389/fspas.2023.1134141
[16] Nayak, U. (2025). AI-Powered Data Pipelines: Leveraging Machine Learning for ETL Optimization. Journal of Software Engineering and Simulation, 11(6), 134-136.
[17] Heffetz, Y., Vainstein, R., & Katz, G. (2019). DeepLine: AutoML Tool for Pipelines Generation using Deep Reinforcement Learning and Hierarchical Actions Filtering. arXiv. https://doi.org/10.48550/arXiv.1911.00061
[18] Chanda, D. (2024). Automated ETL Pipelines for Modern Data Warehousing: Architectures, Challenges, and Emerging Solutions. The Eastasouth Journal of Information System and Computer Science, 1(03), 209–212. https://doi.org/10.58812/esiscs.v1i03.523
[19] Lyashkevych, L., Lyashkevych, V., & Shuvar, R. (2024). Exploratory data analysis possibility in the meaning space using large language models. Electronics and Information Technologies, 1(25), 9, 102–116. http://dx.doi.org/10.30970/eli.25.9
[20] Chen, M., Tworek, J., Jun, H., et al. (2021). Evaluating large language models trained on code. arXiv:2107.03374. https://doi.org/10.48550/arXiv.2107.03374
[21] Wang, Y., Yin, W., Li, B., et al. (2023). CodeT5+: Open code large language models for code understanding and generation. arXiv:2305.07922. https://doi.org/10.48550/arXiv.2305.07922
[22] Rozière, B., Gehring, J., Gloeckle, F., et al. (2023). Code Llama: Open foundation models for code. arXiv:2308.12950. https://doi.org/10.48550/arXiv.2308.12950
[23] Zhang, Y., Wang, C., Xie, T., & Huang, J. (2023). A survey on program synthesis with large language models. arXiv:2311.07989. https://doi.org/10.48550/arXiv.2311.07989
[24] Schick, T., Dwivedi-Yu, J., Dessì, R., Raileanu, R., Lomeli, M., Hambro, E., Zettlemoyer, L., Cancedda, N., & Scialom, T. (2023). Toolformer: Language models can teach themselves to use tools. In Proceedings of the 37th International Conference on Neural Information Processing Systems (Article 2997, pp. 1–13). Curran Associates Inc. https://dl.acm.org/doi/10.5555/3666122.3669119.
[25] Park, J. S., O'Brien, J., Cai, C. J., Morris, M. R., Liang, P., & Bernstein, M. S. (2023, October). Generative agents: Interactive simulacra of human behavior. In Proceedings of the 36th annual acm symposium on user interface software and technology (pp. 1-22). https://doi.org/10.48550/arXiv.2304.03442

СТАЛА ОПТИМІЗАЦІЯ АЛГОРИТМІВ ОБРОБКИ КОНСОЛІДОВАНИХ ДАНИХ НА ОСНОВІ МАШИННОГО НАВЧАННЯ ТА ГЕНЕТИЧНИХ АЛГОРИТМІВ

Василь Ляшкевич
Львівський національний університет імені Івана Франка,
вул. Драгоманова 50, 79005, Львів, Україна

АНОТАЦІЯ

Вступ. Автоматизація побудови аналітичних звітів у промислових компаніях набуває стратегічного значення через різноманіття форматів документів, збільшення обсягів даних та зростання вимог до швидкого формування багатокомпонентних аналітичних матеріалів. Традиційні ETL-конвеєри не справляються зі складністю сучасних інформаційних потоків, особливо коли у процес інтегруються машинне навчання, LLM-моделі та агентні системи. У зв'язку зі швидким прогресом генерації коду та автономних агентів, здатних виконувати складні аналітичні процедури, задача автоматичного конструювання звітних конвеєрів стає все більш перспективною та науково обґрунтованою.

Матеріали та методи. Запропоновано еволюційну модель побудови алгоритмів для опрацювання консолідованих звітів на основі генетичних алгоритмів (ГА). Для генерації звітів, алгоритм визначає конвеєр для побудови візуального компоненту. Популяція визначається як тензор, що забезпечує паралельну еволюцію множини незалежних робочих потоків. Операції класифіковано у чотири групи: ETL, ML, LLM та VIS. Функція пристосованості оцінює сталу довжину конвеєру, покриття ключових типів операцій та їх структурну узгодженість.

Результати. Експериментальні результати показали, що ГА швидко еволюціонує від випадкових NOP-домінованих структур до сталих, логічно узгоджених та функціональних конвеєрів довжиною 10-14 операцій. Найкращі хромосоми сформували три повноцінні візуальні компоненти: прогнозну регресійну модель, семантичну кластеризацію представлень вбудовування та категоріальну діаграму. Така еволюційна закономірність підтверджує, що комбіновані конвеєри можуть будуватися автоматично й адаптивно, а збільшення складності операцій у просторі "смислу", що є векторним простором вбудовування, разом із розвитком генерації коду та агентних архітектур.

Висновки. Запропонована модель демонструє ефективний механізм автоматизованого синтезу звітів з багатьма візуальними компонентами на основі еволюційних конвеєрів. Перспективність методу зростає із розвитком агентних систем ШІ та збільшенням кількості операцій у просторі смислу, що відкриває шлях до повністю автономних систем аналітичної звітності нового покоління.

Ключові слова: консолідовані дані, сталий оптимізаційний підхід, машинне навчання, генетичні алгоритми, LLM, аналітика даних

Received / Одержано	Revised / Доопрацьовано	Accepted / Прийнято	Published / Опубліковано
19 November, 2025	14 December, 2025	14 December, 2025	25 December, 2025