Integrating Backtracking Algorithms and ASCII Steganography for Enhanced Cloud Data Security

Sangeetha Priya B (1), Abith Thomas R A (2), Kumaravelu R (3), Jayatheeratha R (4), Edgin(5)

(1) Assistant Professor, Department of CSE (Internet of Things and Cyber Security Including Blockchain Technology), SNS College Of Engineering, Coimbatore-641107.

(2), (3)(4)(5), Department of CSE (Internet of Things and Cyber Security Including Blockchain

Abstract- Cloud computing offers scalability and accessibility but introduces significant data security challenges, including unauthorized access, data breaches, and ensuring confidentiality in multi-tenant environments. While encryption protects data content, it does not conceal the existence of sensitive information. Steganography provides a complementary layer by hiding data within innocuous cover media. This paper proposes a novel approach to enhance cloud data security by combining ASCII-based steganography with a backtracking algorithm. The backtracking algorithm is employed to optimize the embedding process within ASCII text data (e.g., logs, configuration files, plain text documents commonly found in cloud storage), aiming to maximize embedding capacity and resilience against statistical steganalysis while adhering to specific constraints. We detail the proposed framework, including the data hiding and extraction processes, the role of the backtracking algorithm in selecting optimal embedding locations or parameters based on a defined cost function (e.g., minimizing statistical deviations), and the integration within a cloud storage context. The methodology leverages the subtle manipulation of ASCII characters (e.g., whitespace, control characters, or character substitutions) guided by the backtracking search. We present a theoretical analysis of the security enhancements and discuss potential performance trade-offs. This combined approach aims to provide an additional layer of defense, making sensitive data less conspicuous and harder to detect even if cloud storage is compromised or subjected to surveillance.

Keywords: Cloud Security, Data Hiding, Steganography, ASCII Steganography, Text Steganography, Backtracking Algorithm, Information Security, Data Confidentiality.

I. INTRODUCTION

The proliferation of cloud computing services (IaaS, PaaS, SaaS) has revolutionized data storage, processing, and accessibility [Reference 1]. Organizations leverage the cloud for its cost-effectiveness, scalability, and flexibility. However, entrusting sensitive data to third-party providers raises significant security concerns [Reference 2]. Data breaches, insider threats, insecure APIs, misconfigurations, and inadequate data isolation in multi-tenant architectures pose substantial risks to data confidentiality, integrity, and availability [Reference 3]. Traditional security mechanisms like encryption, access control (IAM), and network security protocols are fundamental but often insufficient on their own. Encryption protects data content from unauthorized viewing *if* the decryption key is secure, but the encrypted data itself remains visible and potentially flags the presence of valuable information, attracting attackers [Reference 4]. Furthermore, certain cloud operations might require data to be decrypted temporarily, creating windows of vulnerability. Steganography, the art and science of hiding information within other, seemingly harmless data (cover media), offers a complementary security layer known as "security through obscurity" [Reference 5]. Unlike cryptography, which makes data unreadable, steganography aims to make the *existence* of the secret data undetectable. By embedding sensitive information within innocuous cover files stored or transmitted in the cloud (e.g., images, audio files, text documents), one can reduce the likelihood of detection. Text steganography, particularly using ASCII-based cover media, is relevant in cloud environments where text files like logs, source code, configuration files, and datasets are abundant [Reference 6]. ASCII steganography techniques often involve subtle modifications like manipulating whitespace (spaces, tabs), using special Unicode characters that look like standard ASCII, or slightly altering character codes [Reference 7]. However, naive embedding can introduce statistical anomalies detectable by steganalysis tools. This paper proposes a novel method to enhance ASCII steganography for cloud data security by incorporating a backtracking algorithm. Backtracking is a general algorithmic technique for solving problems recursively by trying to build a solution incrementally, one piece at a time, removing those solutions ("backtracking") that fail to satisfy the problem's constraints [Reference 8]. We hypothesize that backtracking can be used to systematically explore the potential embedding locations or parameters within an ASCII cover text, guided by constraints designed to minimize statistical detectability and maximize capacity or robustness.

The primary contributions of this paper are:

1. A Novel Framework: Proposing the integration of a backtracking algorithm with ASCII steganography specifically for enhancing cloud data security.
2. Optimized Embedding Strategy: Detailing how backtracking can systematically search for optimal embedding configurations within ASCII text based on predefined security and capacity constraints.
3. Enhanced Security Potential: Analyzing how this approach can potentially improve resilience against statistical steganalysis compared to simpler embedding methods.
4. Cloud Contextualization: Discussing the applicability and integration of the proposed method within typical cloud storage and data processing scenarios.

II. RELATED WORK

This research builds upon existing work in cloud security and steganography.

2.1. Cloud Data Security: Numerous studies have addressed cloud security challenges. Solutions range from advanced cryptographic techniques like homomorphic encryption [Reference 9] and searchable encryption [Reference 10] to robust identity and access management (IAM) policies [Reference 11], intrusion detection systems (IDS) tailored for cloud environments [Reference 12], and data loss prevention (DLP) strategies [Reference 13]. While effective, these primarily focus on preventing unauthorized access or making intercepted data unreadable, not concealing its existence.

2.2. Steganography: Steganography has been extensively studied for various cover media, including images [Reference 14], audio [Reference 15], and video [Reference 16]. These methods often exploit redundancy or noise within the cover medium.

2.3. Text Steganography: Text steganography presents unique challenges due to the lower redundancy in text compared to multimedia files [Reference 6, 7]. Techniques include:

2.4. Format-Based Methods: Manipulating whitespace (spaces, tabs), line shifts, or text justification [Reference 17]. These are often simple but can have low capacity and be vulnerable to reformatting or basic statistical analysis.

2.5. Linguistic Steganography: Modifying text content using synonym substitution, grammatical changes, or generating cover text that naturally encodes the secret message [Reference 18]. This is often more robust but complex and may require sophisticated natural language processing.

2.6. Character/Code-Based Methods: Using visually similar Unicode characters (homoglyphs) or subtly modifying character encodings [Reference 19].

2.7. Steganalysis: Counteracting steganography involves steganalysis, which aims to detect the presence of hidden messages [Reference 20]. For text, statistical analysis (e.g., frequency analysis of characters, words, or whitespace patterns) is a common detection method. Effective steganography must minimize deviations from the expected statistics of the cover medium.

2.8. Optimization Algorithms in Steganography: Some research has explored using optimization algorithms (e.g., genetic algorithms, simulated annealing) to improve steganographic embedding by minimizing distortion or maximizing capacity under certain constraints [Reference 21]. However, the specific application of backtracking algorithms to guide ASCII steganography embedding for cloud security appears relatively unexplored. Backtracking's strength lies in exhaustive, constrained search, which could be beneficial for finding optimal embedding patterns within the discrete structure of text.

Our work differs by specifically combining the systematic, constraint-driven search capabilities of backtracking with the nuances of ASCII steganography, targeting the unique environment and data types prevalent in cloud storage. We aim to leverage backtracking not just for simple optimization but for finding secure embedding configurations that adhere to complex constraints related to statistical invisibility in text data.

III. BACKGROUND CONCEPTS

3.1 Cloud Computing Security Threats Data stored in the cloud faces various threats:

- Unauthorized Access: Due to misconfigured permissions, weak authentication, or exploited vulnerabilities.
- Data Breaches: Malicious actors gaining access to large volumes of data stored by the cloud provider or its tenants.
- Insecure Interfaces/APIs: Vulnerabilities in cloud management APIs can be exploited for unauthorized access or control.
- Multi-tenancy Risks: Shared resources (CPU, memory, storage) potentially allowing interference or data leakage between tenants if isolation mechanisms fail.
- Insider Threats: Malicious actions by employees of the cloud provider or the tenant organization.
- Data Loss/Leakage: Accidental deletion, hardware failures, or subtle leakage through side channels.

3.2 Backtracking Algorithms Backtracking is an algorithmic paradigm that finds solutions to computational problems, notably constraint satisfaction problems, by incrementally building candidate solutions and abandoning ("backtracking") a candidate as soon as it determines that it cannot possibly lead to a valid, complete solution [Reference 8].

Key characteristics:

- State Space Search: Explores a state space tree representing possible solutions.
- Recursive/Iterative Structure: Typically implemented recursively.
- Constraints: Uses constraints to prune branches of the search tree that violate problem conditions.
- Exhaustive Search (within constraints): Can find all possible solutions or the first/optimal one satisfying the criteria.

Example applications include solving puzzles (N-Queens, Sudoku), parsing languages, and combinatorial optimization problems. In our context, the "state" could represent the current state of the cover text after embedding some bits, and the "constraints" relate to maintaining statistical properties or adhering to embedding rules.

3.3 ASCII Steganography ASCII (American Standard Code for Information Interchange) defines character encodings for electronic communication. Standard ASCII uses 7 bits (128 characters), including control characters, punctuation, numbers, and letters. Extended ASCII uses 8 bits (256 characters).

ASCII steganography techniques exploit characteristics of text files:

- Whitespace Manipulation: Adding/removing spaces or tabs at the end of lines, between words, or using different combinations of space/tab characters to encode bits [Reference 17]. This is simple but sensitive to reformatting.
- Character Substitution: Replacing characters with visually similar ones (homoglyphs) from extended character sets (e.g., Unicode) or subtly modifying existing characters (less common with pure ASCII).
- Using Control Characters: Embedding data using non-printable control characters, although this can sometimes corrupt file interpretation.
- LSB of Character Codes: Modifying the least significant bit(s) of the ASCII codes of characters. This can cause subtle, potentially noticeable changes depending on the character.

- Custom Encoding Schemes: Defining specific patterns or sequences of characters/whitespace to represent bits.

The main challenges are achieving sufficient embedding capacity without introducing detectable statistical changes (e.g., unusual whitespace frequency) and ensuring robustness against accidental or intentional modifications (e.g., text editors automatically trimming trailing whitespace).

IV. PROPOSED METHODOLOGY

We propose the Backtracking-Enhanced ASCII Steganography (BACS) framework to securely hide data within ASCII text files stored in a cloud environment. The core idea is to use backtracking to guide the embedding process, optimizing for security (low detectability) and capacity based on defined constraints.

4.1 System Architecture The BACS system operates within a typical cloud storage scenario (e.g., object storage like AWS S3, Azure Blob Storage, Google Cloud Storage).

Input: Secret Message (binary data M), Cover Text (ASCII file C), Stego-Key (K, optional, for added security like selecting specific character sets or initial parameters), Security/Capacity Constraints (P).

- Processing (Cloud Function/VM):

1. Preprocessing: The secret message M may be optionally encrypted and compressed. The cover text C is analyzed to identify potential embedding locations (e.g., end-of-lines, inter-word spaces, specific character positions).
2. Backtracking Embedding Module: This core module uses the backtracking algorithm to embed M into C.
3. Output: Stego-Text (ASCII file S containing the hidden message).

- Storage: The stego-text S is stored in the cloud, appearing as a regular text file.
- Extraction (Cloud Function/VM):

1. Input: Stego-Text S, Stego-Key K (if used).
2. Backtracking Extraction Module: Recovers the hidden bitstream. The backtracking logic might be needed if the embedding path itself needs to be reconstructed based on constraints, or a simpler direct extraction based on the key might suffice depending on the embedding method.
3. Postprocessing: Decompress and decrypt (if applicable) the extracted bitstream to recover the original secret message M.

4.2 ASCII Steganography Technique BACS can be adapted to various ASCII steganography techniques. For illustration, let's consider a whitespace manipulation technique: encoding bits using the number of trailing spaces/tabs at the end of lines.

- '0': Represented by one trailing space.
- '1': Represented by two trailing spaces.
- (Alternative: Use combinations of space/tab for higher capacity).

Constraints (P) could include:

- Maximum number of trailing whitespace characters allowed per line (to avoid suspicion).
- Maintaining the overall whitespace statistics (frequency of lines with 0, 1, 2... trailing spaces) close to the original cover text or a 'natural' distribution.
- Ensuring embedding does not significantly alter file size beyond a certain threshold.
- Skipping lines that are empty or too short/long.

4.3 Backtracking Algorithm Integration The backtracking algorithm searches for an optimal sequence of embedding actions (e.g., which lines to use and how many spaces/tabs to add) that satisfy the constraints P while embedding the entire message M.

- State Representation: A state in the search could be (line_index, bits_embedded, current_stats), representing the current line being considered, the number of secret bits already embedded, and the current statistics of the modified text (e.g., whitespace distribution).

- Search Space: The tree nodes represent choices at each potential embedding location (e.g., embed '0' here, embed '1' here, skip this location).
- Constraint Checking: At each step, the algorithm checks if the potential embedding action violates any constraint in P. For example:
    - Does adding two spaces exceed the max allowed trailing whitespace?
    - Does embedding here push the whitespace frequency distribution outside acceptable bounds (e.g., compared using Chi-squared test against expected distribution)?
- Cost Function (Optional, for Optimization): If aiming for the 'best' embedding (e.g., lowest statistical deviation), a cost function can guide the search. The algorithm would seek a path minimizing this cost. Cost could be a measure of statistical difference between the stego-text and the original cover or a model of natural text.
- Backtracking: If a path leads to a state where constraints are violated or the message cannot be fully embedded using the remaining locations, the algorithm backtracks to the previous decision point and explores alternative choices.

4.4 Extraction Process Extraction involves reading the stego-text S and reversing the embedding process. If the embedding locations and method are solely determined by the key K and the cover text structure, extraction is straightforward. If the backtracking search path itself encodes information or if multiple valid paths exist, the key K might need to guide the extractor to follow the correct sequence of locations identified during embedding, possibly involving a similar (but simpler) traversal or lookup based on the key.

V. IMPLEMENTATION AND EXPERIMENTAL SETUP (PLANNED)

- Implementation: The proposed BACS framework would be implemented using Python, leveraging libraries for text processing and potentially cloud SDKs (like Boto3 for AWS, Azure SDK, Google Cloud Client Libraries) for integration.
- Cloud Environment: Experiments could be simulated locally or deployed using cloud functions (e.g., AWS Lambda, Google Cloud Functions) triggered by file uploads to cloud storage.
- Datasets:
    - *Cover Texts:* A diverse set of ASCII files representative of cloud data (e.g., system logs, configuration files, source code repositories, plain text notes). Files of varying sizes and structures would be used.
    - *Secret Messages:* Sample binary data of different sizes (small keys, medium-sized messages, larger data chunks).
- Metrics:
    - *Embedding Capacity:* Bits embedded per character or per line of the cover text.
    - *Imperceptibility/Security:* Statistical analysis comparing original cover texts and stego-texts. Metrics could include:
        - Frequency distribution of characters (especially whitespace).
        - Chi-squared statistic comparing distributions.
        - Entropy analysis.
        - Detection rate using standard text steganalysis tools (if available and applicable).
    - *Performance:* Time taken for embedding and extraction processes (computation overhead). CPU and memory usage in the cloud environment.
    - *Robustness:* Sensitivity to minor modifications (e.g., automatic whitespace trimming by editors, format conversions).
- Comparison: The performance and security of BACS would be compared against:
    - Simple/naive ASCII steganography (e.g., appending spaces without optimization).

- Other existing text steganography tools or algorithms.

VI. RESULTS AND DISCUSSION (EXPECTED)

We anticipate the following outcomes:

- Feasibility: The experiments should demonstrate the feasibility of embedding and extracting data using the BACS framework.
- Security Enhancement: We expect the backtracking approach, guided by statistical constraints, to produce stego-texts that are significantly harder to detect using statistical steganalysis compared to naive methods. The IsValid function incorporating statistical checks is crucial here. Results would likely show statistical metrics (e.g., Chi-squared values) for BACS stego-texts being closer to those of original cover texts.
- Capacity vs. Security Trade-off: Stricter security constraints (e.g., tighter bounds on statistical deviation) will likely reduce the embedding capacity. The backtracking algorithm allows exploring this trade-off systematically. Results should quantify this relationship.
- Performance Overhead: The backtracking search introduces computational overhead compared to simple embedding. Experiments will measure this overhead, which is expected to increase with the size of the cover text, the message size, and the complexity of the constraints. The efficiency of the IsValid check is critical. Pruning ineffective search paths early is key to managing performance.
- Robustness: The chosen ASCII steganography technique (e.g., trailing whitespace) might have inherent robustness issues. The results should evaluate how well the stego-texts survive common text manipulations. Using techniques less susceptible to automatic formatting might be necessary for practical deployment.

VII. SECURITY ANALYSIS

- Confidentiality: The primary goal is to hide the *existence* of the secret message. By embedding data within seemingly normal ASCII files and using backtracking to minimize statistical anomalies, BACS aims to make the stego-text indistinguishable from benign traffic or stored files, thus enhancing confidentiality. Optional encryption of the message before embedding adds another layer.
- Integrity: Standard steganography does not guarantee the integrity of the hidden message if the stego-text is modified. Error correction codes could be added to the secret message before embedding to provide some resilience against minor modifications. Hash checksums embedded alongside the data could detect tampering upon extraction.
- Detectability (Resistance to Steganalysis): The core security contribution is reduced detectability. The backtracking algorithm's role is to ensure the embedding process adheres to constraints designed to mimic the statistical profile of natural/original text files, making detection via statistical analysis harder. The effectiveness depends on the accuracy of the statistical models and constraints used.
- Key Management: If a stego-key K is used (e.g., to seed the backtracking search, select parameters, or encrypt the message), standard key management practices are essential. The security of the key directly impacts the security of the hidden data.

VIII. Limitations

- Embedding Capacity: Text steganography generally offers lower capacity compared to multimedia steganography. BACS is suitable for hiding smaller amounts of sensitive data (keys, configuration parameters, short messages) rather than bulk data.
- Computational Overhead: The backtracking search can be computationally intensive, potentially making it unsuitable for resource-constrained environments or applications requiring very fast embedding/extraction.
- Sensitivity to Formatting: Techniques like whitespace manipulation are vulnerable to text editors or systems that automatically normalize

whitespace. Robustness requires careful selection of the underlying ASCII technique or additional mechanisms.

- Steganalysis Evolution: Advanced steganalysis techniques, possibly using machine learning, might be developed that can detect the subtle patterns introduced even by optimized embedding.
- Cover Text Dependency: The effectiveness depends on the availability of suitable ASCII cover texts with sufficient redundancy or acceptable locations for modification within the cloud environment.

IX. Future Work

- Adaptive Constraints: Develop methods for dynamically adjusting constraints based on the specific characteristics of the cover text using machine learning.
- Hybrid Approaches: Combine BACS with linguistic steganography features, potentially using backtracking to optimize synonym choices or sentence structures.
- Improved Robustness: Investigate ASCII steganography techniques inherently more robust to reformatting and integrate them into the BACS framework. Incorporate stronger error correction codes.
- Performance Optimization: Explore heuristics or pruning strategies to accelerate the backtracking search without significantly compromising security. Parallelize the search process.
- Formal Security Proofs: Develop more formal models and proofs regarding the security (indistinguishability) provided by BACS under specific statistical assumptions.
- Real-World Deployment & Evaluation: Implement and evaluate BACS in a real cloud environment (e.g., securing configuration secrets, hiding audit logs snippets) to assess practical challenges and effectiveness.

X. Conclusion

This paper proposed BACS, a framework combining ASCII steganography with a backtracking algorithm to enhance data security in cloud environments. By systematically searching for embedding patterns that satisfy predefined security constraints (primarily minimizing statistical detectability), BACS aims to hide sensitive information within common ASCII text files more effectively than naive methods. While traditional encryption protects content, BACS focuses on concealing the existence of secret data, providing a complementary layer of security against surveillance and targeted attacks in the cloud. The backtracking approach allows for a principled exploration of the trade-offs between embedding capacity, security, and computational cost. Although challenges related to capacity, robustness, and computational overhead exist, the proposed methodology presents a promising direction for leveraging algorithmic optimization techniques to improve text steganography for securing data in the increasingly pervasive cloud landscape. Further research, implementation, and rigorous experimental evaluation are needed to fully assess its practical potential and limitations.

REFERENCE

[1] *P. Mell and T. Grance, "The NIST Definition of Cloud Computing," National Institute of Standards and Technology, Special Publication 800-145, Sept. 2011.*

[2] *K. Hashizume, D. G. Rosado, E. Fernández-Medina, and E. B. Fernandez, "An analysis of security issues for cloud computing," Journal of Internet Services and Applications, vol. 4, no. 1, pp. 1-13, Dec. 2013.*

[3] *S. Subashini and V. Kavitha, "A survey on security issues in service delivery models of cloud computing," Journal of Network and Computer Applications, vol. 34, no. 1, pp. 1-11, Jan. 2011.*

[4] *R. A. Popa, C. M. S. Redfield, N. Zeldovich, and H. Balakrishnan, "CryptDB: Protecting confidentiality with encrypted query processing," in Proc. 23rd ACM Symp. Operating Systems Principles (SOSP), 2011, pp. 85–100. (Illustrates complexity and potential vulnerabilities even with encryption).*

[5] *N. F. Johnson and S. Jajodia, "Exploring steganography: Seeing the unseen," Computer, vol. 31, no. 2, pp. 26-34, Feb. 1998.*

[6] *W. Bender, D. Gruhl, N. Morimoto, and A. Lu, "Techniques for data hiding," IBM Systems Journal, vol. 35, no. 3.4, pp. 313-336, 1996. (Classic paper, includes text methods).*

[7] *M. A. A. Mohamed, A. A. M. Rahma, and H. A. H. Al-any, "A survey on text steganography techniques," in Proc. Int. Conf. on Computer and Information Sciences (ICCOINS), 2014, pp.1-6.*

[8] *T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, Introduction to Algorithms, 3rd ed. MIT Press, 2009. (Standard textbook covering backtracking).*

[9] *F. Armknecht, C. Boyd, C. Carr, K. Gjøsteen, A. Jäschke, C. A. Reuter, and T. Strandberg, "A guide to fully homomorphic encryption," IACR Cryptology ePrint Archive, Report 2015/1192, 2015. [Online]. Available: https://eprint.iacr.org/2015/1192*

[10] *S. Kamara and K. Lauter, "Cryptographic cloud storage," in Proc. Int. Conf. on Financial Cryptography and Data Security (FC), 2010, pp. 136-149.*

[11] *M. S. Hasan, E. S. Hosseini, R. K. St-Hilaire, and A. C. Squicciarini, "A survey on identity management in cloud computing," ACM Computing Surveys, vol. 53, no. 6, pp. 1-38, Dec. 2020.*

[12] *C. Modi, D. Patel, B. Borisaniya, H. Patel, A. Patel, and M. Rajarajan, "A survey of intrusion detection techniques in cloud," Journal of Network and Computer Applications, vol. 36, no. 1, pp. 42-57, Jan. 2013.*

[13] *A. A. Ghafoor, M. A. Al-Maitah, and H. M. Al-Maitah, "Data Loss Prevention in Cloud Computing: A Survey," Int. Journal of Computer Network and Information Security (IJCNIS), vol. 12, no. 1, pp. 39-53, Feb. 2020.*

[14] *J. Fridrich, M. Goljan, and R. Du, "Reliable detection of LSB steganography in color and grayscale images," in Proc. ACM Workshop on Multimedia and Security, 2001, pp. 27-30.*

[15] *K. Gopalan, "Audio steganography using bit modification," in Proc. IEEE Int. Conf. on Acoustics, Speech, and Signal Processing (ICASSP), vol. 2, 2003, pp. II-357-II-360.*

[16] *K. A. T. Elshoush and E. A. E. Dahshan, "Video Steganography: A Survey," ACM Computing Surveys, vol. 53, no. 3, pp. 1-36, Jun. 2020.*

[17] *S. H. Low, N. F. Maxemchuk, J. T. Brassil, and L. O'Gorman, "Document marking and identification using both line and word shifting," in Proc. IEEE INFOCOM '95, 1995, pp. 853-860.*

[18] *M. Topkara, G. M. Topkara, and M. J. Atallah, "The hiding virtues of ambiguity: On the limits of linguistic steganography," in Proc. Int. Workshop on Information Hiding (IH), 2006, pp. 164-181.*

[19] *V. L. L. Thing and K. H. K. Teh, "Unicode based Steganography for Instant Messaging Applications," in Proc. IEEE Int. Conf. on Telecommunications and Malaysia Int. Conf. on Communications (ICT-MICC), 2007, pp. 661-665.*

[20] *J. Fridrich and M. Goljan, "Practical steganalysis of digital images: state of the art," in Security and Watermarking of Multimedia Contents IV, Proc. SPIE, vol. 4675, 2002, pp. 1-13. (Focuses on images but outlines general steganalysis principles).*

[21] *C.-K. Chan and L. M. Cheng, "Hiding data in images by simple LSB substitution," Pattern Recognition, vol. 37, no. 3, pp. 469-474, Mar. 2004. (While basic LSB, discusses capacity/distortion trade-offs relevant to optimization)*