aerospace

MDPI

Article

Sparse Single-Use Thruster Selection for Control Moment Tracking Using a Depth-First Branch-and-Bound Algorithm

Ha-min Jeon * and Tae Young Kang

LIG Defense&Aerospace, 333, Pangyo-ro, Bundang-gu, Seongnam-si 13488, Gyeonggi-do, Republic of Korea; taeyoung.kang1@ligdna.com
* Correspondence: hamin.jeon@ligdna.com

Abstract

In high-altitude interception, low atmospheric density limits the effectiveness of aerodynamic control, making thruster-based attitude control essential. In systems using single-use impulse-type lateral thrusters, each actuator can be fired only once, generates a fixed thrust magnitude, and is subject to a limit on the number of simultaneously active thrusters. Therefore, selecting an appropriate set of thrusters to track a desired control moment can be formulated as a cardinality-constrained combinatorial optimization problem. This paper proposes a depth-first search (DFS)-based branch-and-bound algorithm for sparse thruster selection. The objective is to minimize the tracking error between the generated and desired control moments while penalizing the number of active thrusters. To improve computational efficiency, thrusters are ordered by moment magnitude, and a problem-specific lower bound is derived from the residual moment and an upper bound on the achievable contribution of the remaining thrusters. This bound enables effective pruning of unpromising branches. The search space is further reduced by reformulating the problem using symmetric thruster pairs that generate opposing moments. Numerical results show that the proposed method achieves accurate moment tracking while significantly reducing computation time compared with the exact mixed-integer quadratic programming (MIQP) benchmark. Mixed-integer linear programming (MILP) is also included as an additional mixed-integer linear surrogate comparison.

Keywords: single-use impulse thruster; discrete control allocation; moment tracking; branch-and-bound; combinatorial optimization; sparse optimization

Check for updates

Academic Editors: Seokwon Lee and Mingu Kim

Received: 15 March 2026
Revised: 6 May 2026
Accepted: 8 May 2026
Published: 10 May 2026

Copyright: © 2026 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license.

1. Introduction

In high-altitude interception environments, low atmospheric density limits the effectiveness of aerodynamic control, making it difficult for interceptors to achieve sufficient maneuverability using aerodynamic surfaces alone [1–3]. In scenarios requiring rapid attitude changes and high maneuver responsiveness, thruster-based attitude control becomes a practical alternative. In such systems, the control moment generated by the attitude controller must be realized through discrete thruster actuation, which can be naturally interpreted as a control allocation (CA) problem, where a desired control moment is distributed among multiple actuators. Extensive research has been conducted on guidance and control methods using continuously operating lateral thrusters, such as thrust vector control (TVC) and divert and attitude control system (DACS) [4–7]. Accordingly, control allocation techniques for continuous actuators have been widely studied. Botelho et al. [8] proposed an explicit control allocation method based on multiparametric optimization to overcome the

limitations of conventional pseudoinverse and LP/QP-based approaches, enabling real-time onboard implementation using a lookup-table structure while incorporating minimum firing constraints. Kim et al. [9] developed a control allocation framework based on linear programming (LP) and relaxed quadratic programming (QP), solved using a primal–dual interior-point method, and showed that LP formulations may become infeasible under actuator saturation, whereas relaxed QP formulations can provide feasible solutions by allowing for gradual convergence. Deng et al. [10] proposed a QP-based control allocation method for redundant thruster configurations, improving fault tolerance and control performance. However, these approaches are primarily designed for continuously operating actuators and do not explicitly account for the discrete and single-use characteristics of impulse-type thrusters.

In contrast, impulse-type lateral thrusters exhibit fundamentally different characteristics from continuous actuators. These thrusters are typically solid-propellant devices that can be fired only once, generate a fixed thrust magnitude and burn duration, and are subject to a limit on the number of simultaneously active thrusters. Moreover, because the thrusters are discretely distributed around the interceptor body, generating a desired control moment requires selecting an appropriate combination of thrusters from multiple candidates. As a result, the thruster selection problem can be formulated as a discrete, combinatorial optimization problem. Research on control using impulse-type thrusters remains relatively limited. Jitpraphai and Costello [11] proposed a trajectory tracking control method that selects a single thruster based on position error, roll rate, and burn time, firing the thruster whose direction is closest to the required correction force. Burchett and Costello [12] developed a control strategy using short-duration lateral thrust pulses, where a single thruster is selected at each firing instance based on timing and roll attitude. Gao et al. [13] formulated an optimal control problem to compute a firing angle command and selected a single thruster whose direction best matches the required force. Zang et al. [14] formulated the thruster firing problem as a mixed-integer linear programming (MILP) problem to minimize the mismatch between commanded and generated forces, achieving optimality at the cost of increased computational complexity. Jeon et al. [15,16] proposed a guidance and control framework using forward lateral thrusters, employing a greedy-based firing logic to approximate the desired control moments using multiple thrusters. Despite these efforts, existing approaches exhibit inherent limitations. In particular, many prior studies focus on selecting a single thruster per command based on directional matching, rather than determining optimal combinations of multiple simultaneously active thrusters for accurate control moment tracking. Heuristic methods offer low computational cost but do not guarantee optimality, whereas optimization-based methods such as MILP ensure optimal solutions but suffer from rapidly increasing computational complexity as the number of thrusters grows. Consequently, the problem of selecting optimal multi-thruster combinations for control moment tracking remains insufficiently addressed.

Accordingly, the problem addressed in this study is formulated as a discrete control allocation (CA) problem with binary, cardinality-constrained actuators, which fundamentally differs from conventional continuous control allocation approaches. Unlike traditional CA methods that assume continuous actuator inputs, the present problem involves single-use impulse thrusters with binary activation, leading to a combinatorial optimization structure. To solve this problem, a depth-first search (DFS)-based branch-and-bound algorithm is proposed for sparse thruster selection to accurately track desired control moments. The proposed method incorporates a problem-specific lower bound derived from the current residual moment and an upper bound on the achievable contribution of the remaining thrusters, enabling efficient pruning of the combinatorial search space. In addition, moment-based thruster ordering and tailored pruning conditions are introduced to further improve

search efficiency. The problem is also reformulated using symmetric thruster pairs, reducing the effective search dimension. Unlike general-purpose mixed-integer optimization methods that rely on relaxation-based bounds and generic branching strategies, the proposed algorithm directly exploits the structural characteristics of the thruster selection problem through customized bounding, ordering, and pruning mechanisms. Numerical experiments are conducted under various command moment conditions to evaluate the performance of the proposed method. The results are compared with mixed-integer optimization approaches, including MILP and mixed-integer quadratic programming (MIQP), in terms of moment tracking accuracy and computational efficiency.

The main contributions of this study are summarized as follows. First, the thruster selection problem for control moment tracking is formulated as a discrete, cardinality-constrained CA problem, explicitly capturing the binary and single-use characteristics of impulse thrusters that are not addressed in conventional continuous CA approaches. Second, a problem-specific lower bound estimation method is developed based on the residual moment and the maximum achievable contribution of the remaining thrusters, enabling effective pruning within the branch-and-bound framework and significantly reducing the computational complexity of the combinatorial search. Third, for symmetric opposing-thruster layouts, a symmetry-based reformulation is introduced by exploiting opposing thruster pairs, which reduces the effective search dimension and improves computational efficiency without sacrificing optimality for the considered symmetric configuration. Finally, the proposed method achieves sparse thruster selection with substantially improved computational efficiency compared with the exact MIQP benchmark, while MILP is included as an additional mixed-integer linear surrogate comparison. The results maintain accurate tracking of both the magnitude and direction of the desired control moment. The remainder of this paper is organized as follows. Section 2 presents the problem formulation and describes the thruster configuration and moment generation model. Section 3 introduces the proposed thruster selection algorithm. Section 4 provides numerical simulations and performance analysis. Finally, Section 5 concludes the paper.

2. Problem Formulation

2.1. Thruster Selection Optimization Problem

This study considers the problem of selecting a subset of lateral impulse thrusters to track a control moment generated by an attitude controller. In such systems, control moments are produced by firing a subset of thrusters distributed around the interceptor body. Each thruster generates a fixed thrust magnitude when activated, and the resulting moment depends on the geometric configuration between the thruster location and the interceptor center of mass. Given a desired control moment, the objective is to determine a thruster set that minimizes the tracking error.

In high-altitude environments, low atmospheric density limits the effectiveness of aerodynamic control surfaces. Under these conditions, lateral thrusters serve as the primary means of attitude control, making efficient thruster selection essential for effective use of the limited propellant. Since impulse thrusters cannot be reused and operate with a fixed thrust magnitude and burn duration, minimizing unnecessary thruster usage is critical. Because each thruster can either be activated or remain inactive, the decision variables are inherently binary. Accordingly, the thruster selection problem can be formulated as the following binary, cardinality-constrained combinatorial optimization problem:

$$\begin{aligned} &\min_{x} J(x) = \|Ax - M_d\|^2 + \lambda \|x\|_1 \\ &\text{subject to } x_i \in \{0,\ 1\},\ \|x\|_1 \ \leq K \end{aligned} \tag{1}$$

where $A \in R^{2\times N}$ denotes the thruster moment matrix, and $x = [x_1, x_2, \ldots, x_N]^T$ represents the binary thruster activation vector indicating whether each thruster is activated. The vector $M_d \in R^2$ represents the desired control moment generated by the attitude controller. The parameter λ is a weighting coefficient that penalizes excessive thruster usage. The variable N denotes the total number of thrusters, and K represents the maximum number of thrusters that can be activated simultaneously.

The first term of the objective function represents the moment tracking error between the generated and desired control moments. The second term penalizes the number of activated thrusters through the L_1-norm, thereby promoting sparse solutions. This formulation seeks to achieve accurate moment tracking while minimizing unnecessary thruster usage. The constraint $x_i \in \{0, 1\}$ reflects the binary operating characteristic of impulse thrusters, which can either be activated once or remain inactive. The constraint $\|x\|_1 \leq K$ limits the maximum number of simultaneously active thrusters. This constraint captures practical system limitations, as excessive thruster usage may deplete limited propellant and reduce control capability during later phases of engagement. The total control moment (M) generated by the selected thrusters can be expressed as a linear combination of individual thruster moments:

$$M = Ax. \tag{2}$$

2.2. Thruster Configuration and Moment Model

Figure 1 illustrates the lateral thruster configuration mounted on the front section of the interceptor. In the figure, the labels #1, #2, ... indicate the indices of the lateral thrusters, and x_b and z_b denote the body-frame x- and z-axes, respectively. $\vec{F}_{LT,i}$ denotes the force generated by the i-th lateral thruster, r_i represents the axial distance from the center of gravity to the thruster location, and θ_r is the angular position of the thruster measured clockwise from the body z_b-axis. A similar lateral impulse thruster configuration has been employed in hit-to-kill interceptors, such as the PAC-3 MSE, for terminal attitude control [17].

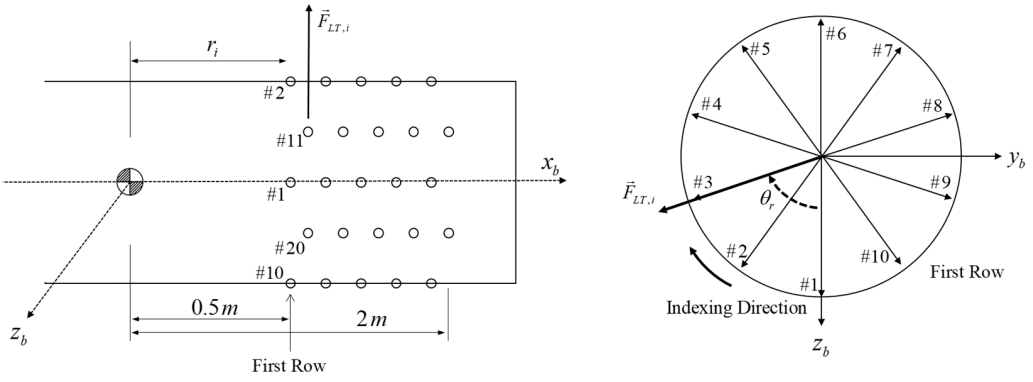


Figure 1. Configuration of front lateral thrusters.

The lateral thrusters are radially arranged around the cylindrical interceptor body. In this study, a total of 100 lateral thrusters are assumed to be distributed along the body. When a thruster is activated, it generates a lateral force perpendicular to the interceptor body axis. The resulting control moment depends on the moment arm between the thruster location and the interceptor center of mass. The control moment generated by the i-th lateral thruster can be expressed as

$$M_i = \begin{bmatrix} l_i & m_i & n_i \end{bmatrix}^T = -F_{LT,i} r_i \begin{bmatrix} 0 & \cos\theta_r & \sin\theta_r \end{bmatrix}^T, \tag{3}$$

where l_i, m_i, and n_i denote the rolling, pitching, and yawing moment components, respectively. Because the lateral thrusters are radially arranged around the interceptor body, no rolling moment is generated (i.e., $l_i = 0$). Therefore, only the pitching and yawing moment components are considered in this study. By stacking the moment components of all thrusters, the thruster moment matrix can be defined as

$$A = \begin{bmatrix} m_1 & m_2 & \dots & m_N \\ n_1 & n_2 & \dots & n_N \end{bmatrix}. \tag{4}$$

Each column of this matrix represents the pitching and yawing moments generated by one thruster. Accordingly, the total control moment for a given thruster selection x follows the linear relation introduced in Equation (2).

3. Thruster Selection Algorithm

3.1. Thruster Selection Problem and Search Framework

In the thruster selection problem, the activation state of each thruster is represented by a binary variable. For a system with N thrusters, the number of possible combinations grows exponentially as 2^N, making exhaustive search computationally prohibitive for real-time applications. To address this combinatorial complexity, heuristic approaches such as greedy algorithms can be employed; however, they do not guarantee global optimality. Alternatively, the problem can be formulated as a mixed-integer optimization problem, which enables systematic exploration of binary combinations. Nevertheless, the computational cost of such approaches increases rapidly with the number of thrusters, limiting their practical applicability.

The control moment generated by a combination of thrusters can be expressed as the linear sum of the moment vectors generated by individual thrusters. This linearity enables the decomposition of the generated moment into two components: the contribution from the currently selected thrusters and the potential contribution from the remaining thrusters. Based on this property, an upper bound on the achievable moment from the remaining thrusters can be conservatively estimated as the sum of the magnitudes of their moment vectors. Using this bound, a lower bound on the achievable moment tracking error at each search node can be derived. This lower bound enables the application of the branch-and-bound framework to efficiently prune search branches that cannot yield improved solutions [18]. Within this framework, the search space can be represented as a binary decision tree, where each level corresponds to a specific thruster. At each level, two branches are considered: selecting the current thruster (select) or excluding it from the current combination (skip). Each node in the tree therefore represents a partial thruster combination defined by the sequence of decisions up to that level.

In this study, the search tree is explored using a depth-first search (DFS) strategy [19], which traverses one branch to its maximum depth before backtracking. In the thruster selection problem, the objective function becomes meaningful only when a complete combination of thrusters is formed, as the generated control moment depends on the combined contribution of multiple thrusters. Consequently, partial selections provide limited information about the final tracking error. An important advantage of DFS is that it can identify a feasible solution early in the search process. Once such a solution is obtained, its cost can be used as the current best value, enabling early pruning of branches that cannot yield improved solutions within the branch-and-bound framework. Efficient pruning relies on estimating a lower bound on the achievable cost at each node, which in this study is computed using the current residual moment and an upper bound on the contribution of the remaining thrusters. To further enhance search efficiency, a priority-

based exploration strategy is introduced, where thrusters are ordered according to the magnitudes of their moment vectors. This ordering increases the likelihood of finding high-quality feasible solutions early, thereby improving pruning effectiveness. By combining DFS traversal, problem-specific lower-bound estimation, and moment-based ordering, the proposed method efficiently explores the search space while systematically pruning unpromising branches and iteratively updating the best feasible solution.

3.2. Lower Bound Estimation

In the branch-and-bound framework, a lower bound on the achievable cost is estimated at each search node to eliminate unnecessary exploration. Conventional approaches typically obtain such bounds by solving relaxed versions of the original problem or by applying general relaxation techniques, often without exploiting the specific problem structure. In contrast, the proposed method derives a problem-specific lower bound by leveraging the characteristics of the thruster selection problem. In particular, the bound is constructed based on the relationship between the residual moment generated by the currently selected thrusters and the maximum achievable contribution from the remaining thrusters. Let k denote the current node in the search tree. The activation state of each thruster is represented by a binary variable $x_i \in \{0, 1\}$, and the set of thrusters selected at node k is defined as follows:

$$S_p(k) = \{i \mid x_i = 1\}. \tag{5}$$

The partial control moment generated by the selected thrusters is given by

$$M_p(k) = \sum_{i \in S_p(k)} a_i, \tag{6}$$

where a_i denotes the moment vector generated by the i-th thruster. The residual moment at node k is defined as

$$r(k) = M_d - M_p(k), \tag{7}$$

where M_d is the desired control moment introduced in Equation (1). Let the set of remaining unselected thrusters be

$$S_r(k) = \{i \mid x_i = 0\}. \tag{8}$$

The additional moment generated by any subset $S \subseteq S_r(k)$ can be expressed as

$$M_r = \sum_{i \in S} a_i. \tag{9}$$

To estimate the maximum achievable moment magnitude from the remaining thrusters, the triangle inequality is applied:

$$\|M_r\| = \left\| \sum_{i \in S} a_i \right\| \leq \sum_{i \in S} \|a_i\| \leq \sum_{i \in S_r(k)} \|a_i\| = R(k), \text{ where } R(k) = \sum_{i \in S_r(k)} \|a_i\|. \tag{10}$$

This bound corresponds to the idealized case in which all remaining thruster moments are perfectly aligned. Although such alignment is unlikely in practice due to the discrete spatial distribution of thrusters, this conservative bound ensures validity during the search process. For computational efficiency, $R(k)$ can be precomputed after sorting thrusters by the magnitudes of their moment vectors. The final residual moment after selecting additional thrusters is given by $r(k) - M_r$. To derive a lower bound on its magnitude, the reverse triangle inequality is used. Starting from

$$r(k) = (r(k) - M_r) + M_r, \tag{11}$$

and applying the triangle inequality yields

$$\|r(k)\| \leq \|r(k) - M_r\| + \|M_r\|. \tag{12}$$

Rearranging gives

$$\|r(k) - M_r\| \geq \|r(k)\| - \|M_r\|. \tag{13}$$

Similarly,

$$\|r(k) - M_r\| \geq \|M_r\| - \|r(k)\|. \tag{14}$$

Combining these results leads to

$$\|r(k) - M_r\| \geq |\|r(k)\| - \|M_r\||. \tag{15}$$

Using $\|M_r\| \leq R(k)$, we obtain

$$\|r(k) - M_r\| \geq \|r(k)\| - R(k). \tag{16}$$

To quantify the minimum achievable residual moment after selecting additional thrusters, we define the minimum residual magnitude as

$$R_{min} = \min_{M_r} \|r(k) - M_r\|. \tag{17}$$

From the inequality in Equation (16), the following bound is obtained:

$$R_{min} \geq \max(0,\ \|r(k)\| - R(k)). \tag{18}$$

The cost function is defined as

$$J(x) = \|Ax - M_d\|^2 + \lambda \|x\|_1. \tag{19}$$

Since $x_i \in \{0,\ 1\}$ and $|x_i| = x_i$, the L_1 norm reduces to

$$\|x\|_1 = \sum_{i=1}^{N} |x_i| = \sum_{i=1}^{N} x_i, \tag{20}$$

which corresponds to the number of activated thrusters. Because the set $S_p(k)$ contains the thrusters already selected at node k, the following bound holds:

$$\|x\|_1 \geq \left|S_p(k)\right| \tag{21}$$

Accordingly, a lower bound on the cost, $J_{LB}(k)$, at node k is given by

$$J_{LB}(k) = (\max(0,\ \|r(k)\| - R(k)))^2 + \lambda \left|S_p(k)\right|. \tag{22}$$

In the Branch-and-Bound process, if $J_{\mathrm{LB}}(k)$ exceeds the best cost found so far, the corresponding branch is pruned. This bound therefore provides a valid lower bound on the achievable cost for all thruster combinations extending from the current node.

3.3. Thruster Ordering Strategy

In DFS-based branch-and-bound search, the order in which nodes are explored significantly affects pruning efficiency. Even with an identical search tree structure, the exploration order determines how quickly a feasible solution is found, which directly im-

pacts pruning effectiveness. In this study, thrusters are prioritized based on the magnitudes of their moment vectors. For each thruster, the following score is defined:

$$\text{score}_i = \|a_i\| \tag{23}$$

The thrusters are sorted in descending order of this score prior to the search. Since thrusters with larger moment magnitudes are more likely to significantly reduce the residual moment, prioritizing them increases the likelihood of achieving substantial reductions at higher levels of the search tree. As a result, high-quality feasible solutions with relatively low objective values can be identified early, improving the effectiveness of pruning in the branch-and-bound process.

The proposed ordering strategy thus serves as a heuristic that guides the DFS exploration toward promising regions of the search space. Importantly, this strategy does not affect the optimality of the branch-and-bound algorithm, as all feasible nodes are still explored unless they are pruned based on the lower-bound condition derived in Section 3.2. Therefore, the ordering only influences the sequence in which nodes are visited, while improving computational efficiency by enabling earlier discovery of high-quality solutions. The overall DFS-based branch-and-bound procedure is summarized in the pseudocode presented in Algorithm 1. Algorithm 1 Pseudocode of DFS-Based Branch-and-Bound Thruster Selection Algorithm.

Algorithm 1: DFS-Based Branch-and-Bound Thruster Selection

Input: A, a_i, M_d, K, λ
Output: S_{best}, J_{best}
Preprocessing:
    Compute moment magnitudes: $\text{score}_i = \|a_i\|$ (Equation (23))
    Sort thrusters in descending order of score_i
    Reorder columns of A accordingly
    Precompute prefix norms: $\text{norm}_{\text{prefix}}(i) \leftarrow \sum_{j=1,\ldots,i} \|a_j\|$
Initialization:
    $S_p \leftarrow []$, $M_p \leftarrow [0\ 0]^T$, $S_{best} \leftarrow []$, $J_{best} \leftarrow \inf$
Start search:
    DFS(1, S_p, M_p)

Procedure DFS(i, S_p, M_p):
    Compute residual: $r \leftarrow M_d - M_p$ (Equation (7))
    Evaluate cost: $J \leftarrow \|r\|^2 + \lambda|S_p|$ (Equation (19))
    Best solution update: if $J < J_{best}$ then $J_{best} \leftarrow J$, $S_{best} \leftarrow S_p$
    Termination check: if $i > N$ *or* $|S_p| \geq K$ then return
    Compute remaining capacity: $\text{remain} \leftarrow \min(K - |S_p|, N - i + 1)$
    Estimate remaining moment bound: $R \leftarrow \sum_{j=i}^{i+\text{remain}-1} \|a_j\|$
    Compute residual norm: $d \leftarrow \|r\|$
    Compute lower bound: $J_{LB} \leftarrow (\max(0,\ d - R))^2 + \lambda|S_p|$ (Equation (22))
    Branch pruning: if $J_{LB} \geq J_{best}$ then return
    Branch (select i): DFS($i+1$, $S_p \cup \{i\}$, $M_p + a_i$)
    Branch (skip i): DFS($i+1$, S_p, M_p)
    return

3.4. Search Space Reduction Using Symmetric Thruster Pairs

The lateral impulse thruster configuration considered in this study includes symmetric thruster pairs that generate opposing moments with respect to the vehicle center. Specifically, for each thruster producing a moment vector a_i, there exists a corresponding thruster that produces the opposite moment $-a_i$. If both thrusters in such a symmetric pair are activated simultaneously, their contributions cancel out, resulting in a net moment of zero. Since this combination does not generate a useful control moment, it can be excluded from the search space without affecting solution optimality. By exploiting this symmetry, each pair of thrusters can be grouped into a single decision unit. In the original formulation, the activation state of each thruster is represented by a binary variable $x_i \in \{0,\ 1\}$. When symmetric pairs are considered, each pair has three possible states: selecting the thruster producing $+a_i$, selecting the thruster producing $-a_i$, or selecting neither. Accordingly, the state of each thruster pair can be represented by a ternary variable

$$s_i \in \{-1, 0, 1\}, \tag{24}$$

where $s_i = 1$ indicates activation of the thruster producing $+a_i$, $s_i = -1$ indicates activation of the thruster producing $-a_i$, and $s_i = 0$ represents the case in which neither thruster in the pair is activated.

This pair-based formulation significantly reduces the size of the search space. In the original formulation, where each thruster is selected independently, the number of possible combinations is 2^N. When symmetric thruster pairs are used as decision units, the number of decision variables becomes $N/2$, and each pair has three possible states. Therefore, the total number of combinations becomes $3^{N/2}$. Since

$$2^N = \left(2^2\right)^{N/2} = 4^{N/2} > 3^{N/2}, \tag{25}$$

the proposed symmetric-pair formulation achieves an exponential reduction in the search space. This reformulation also changes the structure of the search tree. In the original DFS-based approach, each thruster introduces two branches, select or skip, resulting in a binary tree of depth N. In contrast, the pair-based formulation introduces three branches at each node, corresponding to selecting $+a_i$, selecting $-a_i$, or skipping the pair, while reducing the depth of the tree from N to $N/2$. This trade-off leads to a substantial reduction in the overall number of explored nodes.

The proposed DFS-based branch-and-bound algorithm can be directly applied to the pair-based formulation. The lower-bound estimation described in Section 3.2 and the thruster ordering strategy presented in Section 3.3 remain unchanged. Only the branching structure is modified to account for the three possible states of each thruster pair. By exploiting the symmetric structure of the thruster configuration, the effective search dimension is reduced, leading to improved computational efficiency. The symmetric-pair formulation is therefore an optional search-space reduction strategy for opposing thruster layouts. For asymmetric configurations, the original binary branch-and-bound formulation can still be applied without the pair-based reduction.

4. Numerical Simulations

4.1. Simulation Conditions

The performance of the proposed algorithm was evaluated through numerical simulations using the thruster configuration shown in Figure 1. The system consists of a symmetric array of 10×10 thrusters uniformly distributed along the circumferential and axial directions of a cylindrical body. Each thruster generates a fixed thrust magnitude of 3000 N. The axial distance from the center of gravity to each thruster ranges from 0.5 m

to 2 m, resulting in different moment arms depending on the thruster position. Therefore, each thruster produces a different pitching or yawing moment according to its firing direction and axial location. To evaluate the scalability of the proposed method, the maximum number of simultaneously activated thrusters, K, was set to 3, 4, and 5. To investigate the effect of the sparsity penalty, the penalty parameter λ was set to 0, 1000, and 10,000. The case of $\lambda = 0$ corresponds to pure moment tracking without a sparsity penalty. The case of $\lambda = 1000$ represents an intermediate condition where both tracking accuracy and thruster usage affect the solution. The case of $\lambda = 10{,}000$ represents a stronger sparsity-penalty condition, where the objective function places greater weight on reducing the number of activated thrusters. The desired command moment was generated from a physically producible moment. First, a random thruster combination was selected, and the corresponding moment was computed as M_{rand}. The desired command moment was then defined as $M_d = (1 + 1.5\,\text{rand})M_{\text{rand}}$, where rand is a uniformly distributed random number between 0 and 1. This setting preserves the direction of a producible thruster moment while allowing the command magnitude to become larger than the original feasible moment. Therefore, the simulation includes demanding command cases that may not be exactly achievable under the given cardinality constraint.

For each combination of K and λ, 1000 random command moment scenarios were generated. The resulting optimization problems were solved using DFS without ordering, DFS with ordering, Pair DFS, MILP, and MIQP. MILP was solved using MATLAB R2024a intlinprog, whereas MIQP was solved using YALMIP with Gurobi Optimizer 13.0, a widely used commercial solver for mixed-integer optimization. Default solver settings were used for Gurobi to avoid introducing manual solver-specific tuning that could bias the benchmark comparison. The algorithms were compared in terms of computation time, number of explored nodes, pruning ratio, moment tracking error, number of activated thrusters, and objective cost. The number of explored nodes and pruning ratio were evaluated only for the DFS-based algorithms, because these quantities are not applicable to MILP and MIQP. To quantify pruning efficiency, the total number of possible nodes was defined for each DFS framework. For the original DFS-based search, the total number of possible nodes is given by $\sum_{k=0}^{K} \binom{N}{k}$. For the pair-based formulation, the total number of possible nodes is defined as $\sum_{k=0}^{K} \binom{P}{k} 2^k$, where $P = N/2$ denotes the number of symmetric thruster pairs. The factor 2^k accounts for the two possible activation directions within each selected pair. The pruning ratio was defined as the fraction of pruned nodes relative to the total number of possible nodes. All simulations were conducted in MATLAB on a desktop computer equipped with an Intel i7-14700K CPU and 64 GB RAM.

4.2. Simulation Results

The simulation results are summarized in Tables 1–3. Table 1 presents the results for $\lambda = 0$, Table 2 presents the results for $\lambda = 1000$, and Table 3 presents the results for $\lambda = 10,000$. In each table, the values are reported as mean and maximum values over 1000 random command moment scenarios. Overall, the DFS-based methods produced results consistent with MIQP in terms of tracking error, activated thruster count, and objective cost in the tested cases. These results indicate that the proposed DFS-based methods preserved the optimal solution of the original discrete optimization problem in the tested cases. In addition, DFS with ordering reduced the number of explored nodes by finding good feasible solutions earlier, and Pair DFS further reduced the search burden by exploiting the symmetric thruster configuration. The objective cost of MILP is not directly comparable with that of the DFS-based methods and MIQP. The DFS-based methods and MIQP solve the original quadratic objective in Equation (1), whereas MILP solves an absolute-error-based linear surrogate formulation. Therefore, MILP is used as an additional

mixed-integer linear comparison, while MIQP is used as the exact benchmark for the original quadratic formulation. Although MILP can show shorter computation times in some cases, it should be interpreted as a computationally efficient alternative only when an absolute-error tracking criterion is acceptable, rather than as an exact benchmark for the original quadratic objective.

Table 1. Performance comparison for $\lambda = 0$.

K	Algorithm	Mean/Max Time (s)	Mean/Max Explored Nodes	Mean/Max Pruning Ratio (%)	Mean/Max Error	Mean/Max Activated Thrusters	Mean/Max Cost
3	DFS without Ordering	0.013619 /0.054259	78,724 /259,693	52.789 /97.201	699.6046 /13,168.1350	2.9900 /3.0000	5,198,951.668 /173,399,778.1820
	DFS with Ordering	0.002740 /0.010175	15,538 /51,011	90.682 /99.920	699.6046 /13,168.1350	2.9900 /3.0000	5,198,951.668 /173,399,778.1820
	Pair DFS	0.002178 /0.008203	11,132 /34,045	93.120 /99.962	699.6046 /13,168.1350	2.9900 /3.0000	5,198,951.668 /173,399,778.1820
	MILP	0.078058 /0.276981	-[1]	-[1]	742.3509 /13,457.2559	2.9750 /3.0000	837.3298 /15,222.9391
	MIQP	0.573129 /1.176270	-[1]	-[1]	699.6046 /13,168.1350	2.9900 /3.0000	5,198,951.668 /173,399,778.1820
4	DFS without Ordering	0.270128 /1.122321	1,328,624 /5,272,317	67.499 /99.729	577.3000 /10,850.0247	3.9900 /4.0000	4,034,446.812 /117,723,037.0022
	DFS with Ordering	0.042994 /0.142045	210,921 /706,207	94.840 /99.993	577.3000 /10,850.0247	3.9900 /4.0000	4,034,446.812 /117,723,037.0022
	Pair DFS	0.032737 /0.110267	145,629 /496,324	96.214 /99.996	577.3000 /10,850.0247	3.9900 /4.0000	4,034,446.812 /117,723,037.0022
	MILP	0.335266 /0.888447	-[1]	-[1]	606.5051 /11,943.1356	3.9800 /4.0000	697.6940 /13,805.9175
	MIQP	2.405486 /6.807084	-[1]	-[1]	577.3000 /10,850.0247	3.9900 /4.0000	4,034,446.812 /117,723,037.0022
5	DFS without Ordering	2.754878 /16.617334	21,646,116 /83,608,389	72.729 /99.966	453.2777 /15,372.5962	4.9900 /5.0000	4,061,176.338 /236,316,713.2588
	DFS with Ordering	0.381054 /2.080192	2,935,620 /10,497,273	96.302 /99.999	453.2777 /15,372.5962	4.9900 /5.0000	4,061,176.338 /236,316,713.2588
	Pair DFS	0.272827 /1.439412	1,929,945 /6,533,026	97.306 /100.000	453.2777 /15,372.5962	4.9900 /5.0000	4,061,176.338 /236,316,713.2588
	MILP	1.708686 /6.242523	-[1]	-[1]	468.3620 /15,972.7484	4.9700 /5.0000	537.2900 /16,581.5629
	MIQP	19.600070 /76.456848	-[1]	-[1]	453.2777 /15,372.5962	4.9900 /5.0000	4,061,176.338 /236,316,713.2588

[1] The MILP and MIQP solvers do not explicitly report explored nodes or pruning ratios.

Table 2. Performance comparison for $\lambda = 1000$.

K	Algorithm	Mean/Max Time (s)	Mean/Max Explored Nodes	Mean/Max Pruning Ratio (%)	Mean/Max Error	Mean/Max Activated Thrusters	Mean/Max Cost
3	DFS without Ordering	0.009163 /0.037574	68,905 /260,191	58.678 /97.202	866.5465 /11,288.9873	2.9500 /3.0000	5,740,300.430 /127,444,233.5490
	DFS with Ordering	0.001824 /0.007087	13,137 /52,589	92.122 /99.921	866.5465 /11,288.9873	2.9500 /3.0000	5,740,300.430 /127,444,233.5490
	Pair DFS	0.001464 /0.007361	9448 /34,924	94.161 /99.962	866.5465 /11,288.9873	2.9500 /3.0000	5,740,300.430 /127,444,233.5490
	MILP	0.021264 /0.076082	-[1]	-[1]	1076.1492 /11,715.2457	2.1850 /3.0000	3422.4427 /17,311.4025
	MIQP	0.882764 /1.106861	-[1]	-[1]	866.5465 /11,288.9873	2.9500 /3.0000	5,740,300.430 /127,444,233.5490

Table 2. *Cont.*

K	Algorithm	Mean/Max Time (s)	Mean/Max Explored Nodes	Mean/Max Pruning Ratio (%)	Mean/Max Error	Mean/Max Activated Thrusters	Mean/Max Cost
4	DFS without Ordering	0.162853 /0.709939	1,262,645 /5,247,849	69.113 /99.763	586.6548 /11,568.4967	3.6400 /4.0000	4,074,189.751 /133,834,116.0623
	DFS with Ordering	0.026820 /0.094447	207,549 /717,087	94.923 /99.992	586.6548 /11,568.4967	3.6400 /4.0000	4,074,189.751 /133,834,116.0623
	Pair DFS	0.020140 /0.068375	144,172 /499,255	96.252 /99.995	586.6548 /11,568.4967	3.6400 /4.0000	4,074,189.751 /133,834,116.0623
	MILP	0.027436 /0.119933	-[1]	-[1]	843.2481 /11,686.0065	2.5400 /4.0000	3498.8974 /16,511.0501
	MIQP	1.173400 /3.095947	-[1]	-[1]	586.6548 /11,568.4967	3.6400 /4.0000	4,074,189.751 /133,834,116.0623
5	DFS without Ordering	2.127089 /16.132355	15,021,194 /85,378,567	81.076 /99.961	475.7991 /21,568.0006	3.9850 /5.0000	4,405,504.272 /465,183,648.8881
	DFS with Ordering	0.350409 /2.011898	2,377,408 /10,703,049	97.005 /99.999	475.7991 /21,568.0006	3.9850 /5.0000	4,405,504.272 /465,183,648.8881
	Pair DFS	0.250022 /1.383843	1,565,135 /6,621,658	97.815 /100.000	475.7991 /21,568.0006	3.9850 /5.0000	4,405,504.272 /465,183,648.8881
	MILP	0.032860 /0.122616	-[1]	-[1]	745.1580 /23,096.1838	2.7950 /5.0000	3656.8582 /33,375.6635
	MIQP	1.514796 /3.217389	-[1]	-[1]	475.7991 /21,568.0006	3.9850 /5.0000	4,405,504.272 /465,183,648.8881

[1] The MILP and MIQP solvers do not explicitly report explored nodes or pruning ratios.

Table 3. Performance comparison for $\lambda = 10,000$.

K	Algorithm	Mean/Max Time (s)	Mean/Max Explored Nodes	Mean/Max Pruning Ratio (%)	Mean/Max Error	Mean/Max Activated Thrusters	Mean/Max Cost
3	DFS without Ordering	0.009625 /0.034149	75,000 /266,873	55.023 /97.319	1047.1428 /14,213.8874	2.8050 /3.0000	8,050,124.521 /202,064,594.5440
	DFS with Ordering	0.001996 /0.008833	14,747 /55,333	91.156 /99.920	1047.1428 /14,213.8874	2.8050 /3.0000	8,050,124.521 /202,064,594.5440
	Pair DFS	0.001567 /0.010704	10,507 /37,156	93.506 /99.962	1047.1428 /14,213.8874	2.8050 /3.0000	8,050,124.521 /202,064,594.5440
	MILP	0.001863 /0.003670	-[1]	-[1]	11,208.7058 /30,797.4690	0.0000 /0.0000	14,242.0550 /43,541.9267
	MIQP	1.214150 /1.403612	-[1]	-[1]	1047.1428 /14,213.8874	2.8050 /3.0000	8,050,124.521 /202,064,594.5440
4	DFS without Ordering	0.123696 /0.565738	989,556 /4,599,681	75.793 /99.752	531.2547 /15,527.6503	3.2350 /4.0000	3,371,877.816 /241,147,922.3584
	DFS with Ordering	0.021879 /0.089257	170,234 /711,801	95.836 /99.992	531.2547 /15,527.6503	3.2350 /4.0000	3,371,877.816 /241,147,922.3584
	Pair DFS	0.016432 /0.068986	118,502 /495,883	96.919 /99.996	531.2547 /15,527.6503	3.2350 /4.0000	3,371,877.816 /241,147,922.3584
	MILP	0.001915 /0.004030	-[1]	-[1]	13,055.9863 /36,892.5634	0.0000 /0.0000	16,792.2719 /51,935.8392
	MIQP	1.359133 /1.700612	-[1]	-[1]	531.2547 /15,527.6503	3.2350 /4.0000	3,371,877.816 /241,147,922.3584
5	DFS without Ordering	0.912912 /4.178574	6,988,327 /22,572,357	91.196 /99.967	366.5398 /20,661.0462	3.4950 /5.0000	3,716,537.728 /426,928,831.4213
	DFS with Ordering	0.148136 /0.905395	1,100,676 /5,702,477	98.613 /99.999	366.5398 /20,661.0462	3.4950 /5.0000	3,716,537.728 /426,928,831.4213
	Pair DFS	0.108336 /0.589852	747,829 /3,803,377	98.956 /100.000	366.5398 /20,661.0462	3.4950 /5.0000	3,716,537.728 /426,928,831.4213
	MILP	0.002953 /0.190308	-[1]	-[1]	13,339.8663 /46,988.9044	0.0000 /0.0000	17,141.6859 /60,574.0884
	MIQP	1.570937 /2.600032	-[1]	-[1]	366.5398 /20,661.0462	3.4950 /5.0000	3,716,537.728 /426,928,831.4213

[1] The MILP and MIQP solvers do not explicitly report explored nodes or pruning ratios.

4.2.1. Results for $\lambda = 0$

Table 1 summarizes the performance comparison for $\lambda = 0$. Since the sparsity penalty is not included in the objective function, the optimization mainly focuses on minimizing the moment tracking error under the cardinality constraint K. In this subsection, all results are described in terms of mean and maximum values.

For $K = 3$, DFS without ordering required computation times of 0.0136 s and 0.0543 s, explored 7.87×10^4 and 2.60×10^5 nodes, and achieved pruning ratios of 52.79% and 97.20%, respectively. Its tracking errors were 7.00×10^2 and 1.32×10^4, the activated thruster counts were 2.99 and 3, and the costs were 5.20×10^6 and 1.73×10^8, respectively. DFS with ordering required computation times of 0.0027 s and 0.0102 s, explored 1.55×10^4 and 5.10×10^4 nodes, and achieved pruning ratios of 90.68% and 99.92%, respectively. Its tracking errors were 7.00×10^2 and 1.32×10^4, the activated thruster counts were 2.99 and 3, and the costs were 5.20×10^6 and 1.73×10^8, respectively. Pair DFS required computation times of 0.0022 s and 0.0082 s, explored 1.11×10^4 and 3.40×10^4 nodes, and achieved pruning ratios of 93.12% and 99.96%, respectively. Its tracking errors were 7.00×10^2 and 1.32×10^4, the activated thruster counts were 2.99 and 3, and the costs were 5.20×10^6 and 1.73×10^8, respectively. MILP required computation times of 0.0781 s and 0.2770 s, with tracking errors of 7.42×10^2 and 1.35×10^4, activated thruster counts of 2.98 and 3, and costs of 8.37×10^2 and 1.52×10^4, respectively. MIQP required computation times of 0.5731 s and 1.1763 s, with tracking errors of 7.00×10^2 and 1.32×10^4, activated thruster counts of 2.99 and 3, and costs of 5.20×10^6 and 1.73×10^8, respectively.

For $K = 4$, DFS without ordering required computation times of 0.2701 s and 1.1223 s, explored 1.33×10^6 and 5.27×10^6 nodes, and achieved pruning ratios of 67.50% and 99.73%, respectively. Its tracking errors were 5.77×10^2 and 1.09×10^4, the activated thruster counts were 3.99 and 4, and the costs were 4.03×10^6 and 1.18×10^8, respectively. DFS with ordering required computation times of 0.0430 s and 0.1420 s, explored 2.11×10^5 and 7.06×10^5 nodes, and achieved pruning ratios of 94.84% and 99.99%, respectively. Its tracking errors were 5.77×10^2 and 1.09×10^4, the activated thruster counts were 3.99 and 4, and the costs were 4.03×10^6 and 1.18×10^8, respectively. Pair DFS required computation times of 0.0327 s and 0.1103 s, explored 1.46×10^5 and 4.96×10^5 nodes, and achieved pruning ratios of 96.21% and 99.996%, respectively. Its tracking errors were 5.77×10^2 and 1.09×10^4, the activated thruster counts were 3.99 and 4, and the costs were 4.03×10^6 and 1.18×10^8, respectively. MILP required computation times of 0.3353 s and 0.8884 s, with tracking errors of 6.07×10^2 and 1.19×10^4, activated thruster counts of 3.98 and 4, and costs of 6.98×10^2 and 1.38×10^4, respectively. MIQP required computation times of 2.4055 s and 6.8071 s, with tracking errors of 5.77×10^2 and 1.09×10^4, activated thruster counts of 3.99 and 4, and costs of 4.03×10^6 and 1.18×10^8, respectively.

For $K = 5$, DFS without ordering required computation times of 2.7549 s and 16.6173 s, explored 2.16×10^7 and 8.36×10^7 nodes, and achieved pruning ratios of 72.73% and 99.97%, respectively. Its tracking errors were 4.53×10^2 and 1.54×10^4, the activated thruster counts were 4.99 and 5, and the costs were 4.06×10^6 and 2.36×10^8, respectively. DFS with ordering required computation times of 0.3811 s and 2.0802 s, explored 2.94×10^6 and 1.05×10^7 nodes, and achieved pruning ratios of 96.30% and 99.999%, respectively. Its tracking errors were 4.53×10^2 and 1.54×10^4, the activated thruster counts were 4.99 and 5, and the costs were 4.06×10^6 and 2.36×10^8, respectively. Pair DFS required computation times of 0.2728 s and 1.4394 s, explored 1.93×10^6 and 6.53×10^6 nodes, and achieved pruning ratios of 97.31% and 100.00%, respectively. Its tracking errors were 4.53×10^2 and 1.54×10^4, the activated thruster counts were 4.99 and 5, and the costs were 4.06×10^6 and 2.36×10^8, respectively. MILP required computation times of 1.7087 s and 6.2425 s, with tracking errors of 4.68×10^2 and 1.60×10^4, activated thruster counts of 4.97 and 5,

and costs of 5.37×10^2 and 1.66×10^4, respectively. MIQP required computation times of 19.6001 s and 76.4568 s, with tracking errors of 4.53×10^2 and 1.54×10^4, activated thruster counts of 4.99 and 5, and costs of 4.06×10^6 and 2.36×10^8, respectively.

For all K values, the DFS-based methods produced the same tracking errors, activated thruster counts, and objective costs as MIQP, confirming that the proposed methods preserved the optimal solution of the original quadratic formulation in the tested cases. As K increased from 3 to 5, the mean computation times of DFS without ordering were 0.0136 s, 0.2701 s, and 2.7549 s, whereas those of DFS with ordering were 0.0027 s, 0.0430 s, and 0.3811 s, and those of Pair DFS were 0.0022 s, 0.0327 s, and 0.2728 s. The mean numbers of explored nodes were 7.87×10^4, 1.33×10^6, and 2.16×10^7 for DFS without ordering, 1.55×10^4, 2.11×10^5, and 2.94×10^6 for DFS with ordering, and 1.11×10^4, 1.46×10^5, and 1.93×10^6 for Pair DFS. The mean pruning ratios were 52.79%, 67.50%, and 72.73% for DFS without ordering, 90.68%, 94.84%, and 96.30% for DFS with ordering, and 93.12%, 96.21%, and 97.31% for Pair DFS. These results show that ordering provides the dominant reduction in search effort, while the pair-based formulation gives an additional improvement by exploiting the symmetric thruster configuration.

In the $\lambda = 0$ case, MILP produced tracking errors that were relatively close to those of the DFS-based methods and MIQP. However, its objective cost was not directly comparable with those of DFS and MIQP because MILP solves an absolute-error-based linear surrogate rather than the original quadratic objective. Therefore, MILP should not be interpreted as an exact optimal benchmark, and MIQP is used as the reference solution for the original quadratic formulation.

4.2.2. Results for $\lambda = 1000$

Table 2 summarizes the performance comparison for $\lambda = 1000$. Compared with $\lambda = 0$, the sparsity penalty is included in the objective function. Therefore, the optimization considers both moment tracking accuracy and the number of activated thrusters. In this subsection, all results are described in terms of mean and maximum values.

For $K = 3$, DFS without ordering required computation times of 0.0092 s and 0.0376 s, explored 6.89×10^4 and 2.60×10^5 nodes, and achieved pruning ratios of 58.68% and 97.20%, respectively. Its tracking errors were 8.67×10^2 and 1.13×10^4, the activated thruster counts were 2.95 and 3, and the costs were 5.74×10^6 and 1.27×10^8, respectively. DFS with ordering required computation times of 0.0018 s and 0.0071 s, explored 1.31×10^4 and 5.26×10^4 nodes, and achieved pruning ratios of 92.12% and 99.92%, respectively. Its tracking errors were 8.67×10^2 and 1.13×10^4, the activated thruster counts were 2.95 and 3, and the costs were 5.74×10^6 and 1.27×10^8, respectively. Pair DFS required computation times of 0.0015 s and 0.0074 s, explored 9.45×10^3 and 3.49×10^4 nodes, and achieved pruning ratios of 94.16% and 99.96%, respectively. Its tracking errors were 8.67×10^2 and 1.13×10^4, the activated thruster counts were 2.95 and 3, and the costs were 5.74×10^6 and 1.27×10^8, respectively. MILP required computation times of 0.0213 s and 0.0761 s, with tracking errors of 1.08×10^3 and 1.17×10^4, activated thruster counts of 2.19 and 3, and costs of 3.42×10^3 and 1.73×10^4, respectively. MIQP required computation times of 0.8828 s and 1.1069 s, with tracking errors of 8.67×10^2 and 1.13×10^4, activated thruster counts of 2.95 and 3, and costs of 5.74×10^6 and 1.27×10^8, respectively.

For $K = 4$, DFS without ordering required computation times of 0.1629 s and 0.7099 s, explored 1.26×10^6 and 5.25×10^6 nodes, and achieved pruning ratios of 69.11% and 99.76%, respectively. Its tracking errors were 5.87×10^2 and 1.16×10^4, the activated thruster counts were 3.64 and 4, and the costs were 4.07×10^6 and 1.34×10^8, respectively. DFS with ordering required computation times of 0.0268 s and 0.0944 s, explored 2.08×10^5 and 7.17×10^5 nodes, and achieved pruning ratios of 94.92% and 99.99%, respectively. Its

tracking errors were 5.87×10^2 and 1.16×10^4, the activated thruster counts were 3.64 and 4, and the costs were 4.07×10^6 and 1.34×10^8, respectively. Pair DFS required computation times of 0.0201 s and 0.0684 s, explored 1.44×10^5 and 4.99×10^5 nodes, and achieved pruning ratios of 96.25% and 99.995%, respectively. Its tracking errors were 5.87×10^2 and 1.16×10^4, the activated thruster counts were 3.64 and 4, and the costs were 4.07×10^6 and 1.34×10^8, respectively. MILP required computation times of 0.0274 s and 0.1199 s, with tracking errors of 8.43×10^2 and 1.17×10^4, activated thruster counts of 2.54 and 4, and costs of 3.50×10^3 and 1.65×10^4, respectively. MIQP required computation times of 1.1734 s and 3.0959 s, with tracking errors of 5.87×10^2 and 1.16×10^4, activated thruster counts of 3.64 and 4, and costs of 4.07×10^6 and 1.34×10^8, respectively.

For $K = 5$, DFS without ordering required computation times of 2.1271 s and 16.1324 s, explored 1.50×10^7 and 8.54×10^7 nodes, and achieved pruning ratios of 81.08% and 99.96%, respectively. Its tracking errors were 4.76×10^2 and 2.16×10^4, the activated thruster counts were 3.99 and 5, and the costs were 4.41×10^6 and 4.65×10^8, respectively. DFS with ordering required computation times of 0.3504 s and 2.0119 s, explored 2.38×10^6 and 1.07×10^7 nodes, and achieved pruning ratios of 97.01% and 99.999%, respectively. Its tracking errors were 4.76×10^2 and 2.16×10^4, the activated thruster counts were 3.99 and 5, and the costs were 4.41×10^6 and 4.65×10^8, respectively. Pair DFS required computation times of 0.2500 s and 1.3838 s, explored 1.57×10^6 and 6.62×10^6 nodes, and achieved pruning ratios of 97.82% and 100.00%, respectively. Its tracking errors were 4.76×10^2 and 2.16×10^4, the activated thruster counts were 3.99 and 5, and the costs were 4.41×10^6 and 4.65×10^8, respectively. MILP required computation times of 0.0329 s and 0.1226 s, with tracking errors of 7.45×10^2 and 2.31×10^4, activated thruster counts of 2.80 and 5, and costs of 3.66×10^3 and 3.34×10^4, respectively. MIQP required computation times of 1.5148 s and 3.2174 s, with tracking errors of 4.76×10^2 and 2.16×10^4, activated thruster counts of 3.99 and 5, and costs of 4.41×10^6 and 4.65×10^8, respectively.

For all K values, the DFS-based methods produced the same tracking errors, activated thruster counts, and objective costs as MIQP, confirming that the proposed methods preserved the optimal solution of the original quadratic formulation in the tested cases. As K increased from 3 to 5, the mean computation times of DFS without ordering were 0.0092 s, 0.1629 s, and 2.1271 s, whereas those of DFS with ordering were 0.0018 s, 0.0268 s, and 0.3504 s, and those of Pair DFS were 0.0015 s, 0.0201 s, and 0.2500 s. The mean numbers of explored nodes were 6.89×10^4, 1.26×10^6, and 1.50×10^7 for DFS without ordering, 1.31×10^4, 2.08×10^5, and 2.38×10^6 for DFS with ordering, and 9.45×10^3, 1.44×10^5, and 1.57×10^6 for Pair DFS. The mean pruning ratios were 58.68%, 69.11%, and 81.08% for DFS without ordering, 92.12%, 94.92%, and 97.01% for DFS with ordering, and 94.16%, 96.25%, and 97.82% for Pair DFS. These results show that the ordering strategy and the pair-based formulation maintained the same trend as in the $\lambda = 0$ case, with ordering providing the dominant reduction in search effort and Pair DFS providing an additional improvement by exploiting the symmetric thruster configuration.

4.2.3. Results for $\lambda = 10,000$

Table 3 summarizes the performance comparison for $\lambda = 10,000$. This case represents a stronger sparsity-penalty condition than $\lambda = 0$ and $\lambda = 1000$. Therefore, the objective function places greater weight on reducing the number of activated thrusters. In this subsection, all results are described in terms of mean and maximum values.

For $K = 3$, DFS without ordering required computation times of 0.0096 s and 0.0341 s, explored 7.50×10^4 and 2.67×10^5 nodes, and achieved pruning ratios of 55.02% and 97.32%, respectively. Its tracking errors were 1.05×10^3 and 1.42×10^4, the activated thruster counts were 2.81 and 3, and the costs were 8.05×10^6 and 2.02×10^8, respectively.

DFS with ordering required computation times of 0.0020 s and 0.0088 s, explored 1.47×10^4 and 5.53×10^4 nodes, and achieved pruning ratios of 91.16% and 99.92%, respectively. Its tracking errors were 1.05×10^3 and 1.42×10^4, the activated thruster counts were 2.81 and 3, and the costs were 8.05×10^6 and 2.02×10^8, respectively. Pair DFS required computation times of 0.0016 s and 0.0107 s, explored 1.05×10^4 and 3.72×10^4 nodes, and achieved pruning ratios of 93.51% and 99.96%, respectively. Its tracking errors were 1.05×10^3 and 1.42×10^4, the activated thruster counts were 2.81 and 3, and the costs were 8.05×10^6 and 2.02×10^8, respectively. MILP required computation times of 0.0019 s and 0.0037 s, with tracking errors of 1.12×10^4 and 3.08×10^4, activated thruster counts of 0 and 0, and costs of 1.42×10^4 and 4.35×10^4, respectively. MIQP required computation times of 1.2142 s and 1.4036 s, with tracking errors of 1.05×10^3 and 1.42×10^4, activated thruster counts of 2.81 and 3, and costs of 8.05×10^6 and 2.02×10^8, respectively.

For $K = 4$, DFS without ordering required computation times of 0.1237 s and 0.5657 s, explored 9.90×10^5 and 4.60×10^6 nodes, and achieved pruning ratios of 75.79% and 99.75%, respectively. Its tracking errors were 5.31×10^2 and 1.55×10^4, the activated thruster counts were 3.24 and 4, and the costs were 3.37×10^6 and 2.41×10^8, respectively. DFS with ordering required computation times of 0.0219 s and 0.0893 s, explored 1.70×10^5 and 7.12×10^5 nodes, and achieved pruning ratios of 95.84% and 99.99%, respectively. Its tracking errors were 5.31×10^2 and 1.55×10^4, the activated thruster counts were 3.24 and 4, and the costs were 3.37×10^6 and 2.41×10^8, respectively. Pair DFS required computation times of 0.0164 s and 0.0690 s, explored 1.19×10^5 and 4.96×10^5 nodes, and achieved pruning ratios of 96.92% and 99.996%, respectively. Its tracking errors were 5.31×10^2 and 1.55×10^4, the activated thruster counts were 3.24 and 4, and the costs were 3.37×10^6 and 2.41×10^8, respectively. MILP required computation times of 0.0019 s and 0.0040 s, with tracking errors of 1.31×10^4 and 3.69×10^4, activated thruster counts of 0 and 0, and costs of 1.68×10^4 and 5.19×10^4, respectively. MIQP required computation times of 1.3591 s and 1.7006 s, with tracking errors of 5.31×10^2 and 1.55×10^4, activated thruster counts of 3.24 and 4, and costs of 3.37×10^6 and 2.41×10^8, respectively.

For $K = 5$, DFS without ordering required computation times of 0.9129 s and 4.1786 s, explored 6.99×10^6 and 2.26×10^7 nodes, and achieved pruning ratios of 91.20% and 99.97%, respectively. Its tracking errors were 3.67×10^2 and 2.07×10^4, the activated thruster counts were 3.50 and 5, and the costs were 3.72×10^6 and 4.27×10^8, respectively. DFS with ordering required computation times of 0.1481 s and 0.9054 s, explored 1.10×10^6 and 5.70×10^6 nodes, and achieved pruning ratios of 98.61% and 99.999%, respectively. Its tracking errors were 3.67×10^2 and 2.07×10^4, the activated thruster counts were 3.50 and 5, and the costs were 3.72×10^6 and 4.27×10^8, respectively. Pair DFS required computation times of 0.1083 s and 0.5899 s, explored 7.48×10^5 and 3.80×10^6 nodes, and achieved pruning ratios of 98.96% and 100.00%, respectively. Its tracking errors were 3.67×10^2 and 2.07×10^4, the activated thruster counts were 3.50 and 5, and the costs were 3.72×10^6 and 4.27×10^8, respectively. MILP required computation times of 0.0030 s and 0.1903 s, with tracking errors of 1.33×10^4 and 4.70×10^4, activated thruster counts of 0 and 0, and costs of 1.71×10^4 and 6.06×10^4, respectively. MIQP required computation times of 1.5709 s and 2.6000 s, with tracking errors of 3.67×10^2 and 2.07×10^4, activated thruster counts of 3.50 and 5, and costs of 3.72×10^6 and 4.27×10^8, respectively.

For all K values, the DFS-based methods produced the same tracking errors, activated thruster counts, and objective costs as MIQP, confirming that the proposed methods preserved the optimal solution of the original quadratic formulation in the tested cases. As K increased from 3 to 5, the mean computation times of DFS without ordering were 0.0096 s, 0.1237 s, and 0.9129 s, whereas those of DFS with ordering were 0.0020 s, 0.0219 s, and 0.1481 s, and those of Pair DFS were 0.0016 s, 0.0164 s, and 0.1083 s. The mean numbers

of explored nodes were 7.50×10^4, 9.90×10^5, and 6.99×10^6 for DFS without ordering, 1.47×10^4, 1.70×10^5, and 1.10×10^6 for DFS with ordering, and 1.05×10^4, 1.19×10^5, and 7.48×10^5 for Pair DFS. The mean pruning ratios were 55.02%, 75.79%, and 91.20% for DFS without ordering, 91.16%, 95.84%, and 98.61% for DFS with ordering, and 93.51%, 96.92%, and 98.96% for Pair DFS. These results show that the ordering strategy and the pair-based formulation maintained the same trend as in the $\lambda = 0$ and $\lambda = 1000$ cases, with ordering providing the dominant reduction in search effort and Pair DFS providing an additional improvement by exploiting the symmetric thruster configuration.

The limitation of the MILP formulation became more pronounced as λ increased. In the $\lambda = 1000$ case, MILP selected fewer thrusters than DFS and MIQP. The mean numbers of activated thrusters obtained by MILP were 2.19, 2.54, and 2.80 for $K = 3$, $K = 4$, and $K = 5$, respectively, whereas DFS and MIQP selected 2.95, 3.64, and 3.99 thrusters on average. This difference became more significant for $\lambda = 10,000$. MILP selected no thrusters for all tested K values, with mean and maximum activated thruster counts of 0 and 0 for $K = 3$, $K = 4$, and $K = 5$. As a result, the tracking errors of MILP were much larger than those of DFS and MIQP. For $K = 3$, the mean tracking error increased from 1.05×10^3 for DFS and MIQP to 1.12×10^4 for MILP. For $K = 4$, it increased from 5.31×10^2 to 1.31×10^4, and for $K = 5$, it increased from 3.67×10^2 to 1.33×10^4. Although MILP showed very short computation times in these cases, this result was mainly caused by selecting the zero-thruster solution rather than by solving an equivalent form of the original quadratic optimization problem. Therefore, the MILP formulation should be interpreted as a linear surrogate comparison, while MIQP is used as the exact mixed-integer benchmark.

4.2.4. Comparison of Activated Thrusters with Respect to λ

This section compares the number of activated thrusters for different values of the sparsity penalty parameter λ. Since DFS without ordering, DFS with ordering, Pair DFS, and MIQP produced almost identical activated thruster counts for the original quadratic formulation, the following discussion focuses on the common optimal results obtained from these methods. For $K = 3$, the mean and maximum numbers of activated thrusters were 2.99 and 3, respectively, when $\lambda = 0$. When $\lambda = 1000$, the mean number of activated thrusters decreased to 2.95, while the maximum value remained 3. When $\lambda = 10,000$, the mean number further decreased to 2.81, and the maximum value remained 3. Therefore, increasing λ from 0 to 1000 and 10,000 gradually reduced the average thruster usage from 2.99 to 2.95 and 2.81 for $K = 3$.

For $K = 4$, the mean and maximum numbers of activated thrusters were 3.99 and 4, respectively, when $\lambda = 0$. When $\lambda = 1000$, the mean number decreased to 3.64, while the maximum value remained 4. When $\lambda = 10,000$, the mean number further decreased to 3.24, and the maximum value remained 4. Thus, for $K = 4$, increasing λ reduced the average thruster usage from 3.99 to 3.64 and 3.24.

For $K = 5$, the mean and maximum numbers of activated thrusters were 4.99 and 5, respectively, when $\lambda = 0$. When $\lambda = 1000$, the mean number decreased to 3.99, while the maximum value remained 5. When $\lambda = 10,000$, the mean number further decreased to 3.50, and the maximum value remained 5. Therefore, for $K = 5$, increasing λ reduced the average thruster usage from 4.99 to 3.99 and 3.50.

These results show that the sparsity penalty effectively reduces the average number of activated thrusters. The reduction becomes more apparent as K increases. For $K = 3$, the mean number of activated thrusters decreased by 0.18 when λ increased from 0 to 10,000. For $K = 4$, the reduction was 0.75, and for $K = 5$, the reduction was 1.49. This indicates that the sparsity penalty has a stronger effect when a larger number of thrusters can be simultaneously activated. However, the maximum number of activated thrusters remained

equal to K in all cases, indicating that the optimizer still selected the maximum allowable number of thrusters for some command moment scenarios when required for tracking performance. In summary, increasing λ shifts the solution preference from using the maximum allowable number of thrusters toward selecting fewer thrusters. This confirms that the penalty term in the objective function effectively controls the trade-off between moment tracking accuracy and thruster usage.

5. Conclusions

This paper proposed a depth-first search (DFS)-based branch-and-bound algorithm for the thruster selection problem in control moment tracking using single-use impulse-type lateral thrusters. Due to the binary and single-use characteristics of impulse thrusters, the problem is formulated as a combinatorial optimization problem with a large search space and cardinality constraints. To address this challenge, a problem-specific lower bound based on the residual moment and the maximum achievable contribution from the remaining thrusters was derived to enable efficient pruning within the branch-and-bound framework. In addition, a moment magnitude-based ordering strategy was introduced to accelerate the discovery of high-quality feasible solutions, and a symmetric thruster-pair formulation was developed to reduce the effective search dimension.

Numerical results demonstrated that the proposed method achieves the same moment tracking performance as the exact MIQP benchmark while significantly reducing computational cost. MILP was also included as an additional mixed-integer linear surrogate comparison, and its results showed that the linear surrogate formulation is not always equivalent to the original quadratic objective. In particular, although DFS without ordering yields the same optimal solution, it incurs substantially higher computational cost, whereas the proposed ordering strategy improves pruning efficiency and accelerates the search process. The symmetric-pair formulation further reduces the number of explored nodes, leading to additional gains in computational efficiency. The sparsity penalty parameter λ was also shown to provide an effective trade-off between tracking accuracy and thruster usage. In summary, the proposed approach provides an efficient and structure-exploiting solution to the discrete control allocation problem for impulse thruster systems and provides a basis for future real-time implementation studies. Future work will focus on processor-level worst-case execution time analysis and real-time implementation on embedded processors. Although maximum computation times were reported in the numerical simulations, further validation is required before applying the proposed algorithm to safety-critical onboard systems.

Author Contributions: Conceptualization, H.-m.J.; methodology, H.-m.J. and T.Y.K.; software, H.-m.J.; validation, H.-m.J. and T.Y.K.; formal analysis, H.-m.J. and T.Y.K.; investigation, H.-m.J.; resources, H.-m.J.; data curation, H.-m.J.; writing—original draft preparation, H.-m.J.; writing—review and editing, H.-m.J. and T.Y.K.; visualization, H.-m.J. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: The data supporting the findings of this study are not publicly available due to confidentiality and proprietary restrictions.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Lee, C.-H.; Kim, T.-H.; Jun, B.-E. Nonlinear acceleration controller design for DACS type kill vehicle. *J. Korean Soc. Propuls. Eng.* **2015**, *19*, 54–64. [CrossRef]
2. Ann, S.; Lee, S.; Kim, Y. Intercept point prediction for midcourse guidance of anti-ballistic missile. In Proceedings of the 26th Mediterranean Conference on Control and Automation, Zadar, Croatia, 19–22 June 2018; pp. 1–9. [CrossRef]
3. Park, H.R. The ballistic missile defense construction strategies of South Korea and Japan: Self-reliance versus cooperation with the US. *J. Int. Area Stud.* **2018**, *25*, 87–105. [CrossRef]
4. Jeon, H.-M.; Park, J.; Ryoo, C.-K. High-altitude terminal guidance and control loop design using thrust vector control. *J. Korean Soc. Aeronaut. Space Sci.* **2022**, *50*, 393–400. [CrossRef]
5. Tournes, C.; Shtessel, Y.; Shkolnikov, I. Autopilot for missiles steered by aerodynamic lift and divert thrusters using nonlinear dynamic sliding manifolds. In Proceedings of the AIAA Guidance, Navigation, and Control Conference, San Francisco, CA, USA, 15–18 August 2005. [CrossRef]
6. Lim, Y.; Lee, W.; Bang, H.; Lee, H. Thrust distribution for attitude control in a variable thrust propulsion system with four ACS nozzles. *Adv. Space Res.* **2017**, *59*, 1848–1860. [CrossRef]
7. Lim, Y.; Lee, J.; Bang, H.; Huh, H.; Lee, H. Reduced actuator set for pressure control and thrust distribution for multinozzle propulsion systems. *Int. J. Aerosp. Eng.* **2017**, *2017*, 4104212. [CrossRef]
8. Botelho, A.; Rosa, P.; Lemos, J.M. Explicit spacecraft thruster control allocation with minimum impulse bit. *IEEE Trans. Control Syst. Technol.* **2025**, *33*, 833–844. [CrossRef]
9. Kim, J.; Kim, J.-H. Spacecraft attitude control with on-off thrusters via convex optimization based control allocation. *Int. J. Aeronaut. Space Sci.* **2025**, *26*, 2695–2711. [CrossRef]
10. Deng, L.; Tao, J. Thrust allocation control of an underwater vehicle with a redundant thruster configuration. *Mathematics* **2025**, *13*, 1766. [CrossRef]
11. Jitpraphai, T.; Costello, M. Dispersion reduction of a direct fire rocket using lateral pulse jets. *J. Spacecr. Rocket.* **2001**, *38*, 929–936. [CrossRef] [PubMed]
12. Burchett, B.; Costello, M. Model predictive lateral pulse jet control of an atmospheric rocket. *J. Guid. Control Dyn.* **2002**, *25*, 860–867. [CrossRef] [PubMed]
13. Gao, M.; Zhang, Y.; Yang, S. Firing control optimization of impulse thrusters for trajectory correction projectiles. *Int. J. Aerosp. Eng.* **2015**, *2015*, 781472. [CrossRef]
14. Zang, Y.; Yang, B.; Huo, X.; Chen, S. Firing logic design for ACS of missile with pulse jets. In Proceedings of the Chinese Control and Decision Conference, Yinchuan, China, 28–30 May 2016; pp. 5596–5601. [CrossRef]
15. Jeon, H.-M.; Kang, T.Y.; Park, J.; Ryoo, C.-K. High-altitude terminal control of anti-air missiles with a terminal booster and front lateral impulse thrusters. *IEEE Trans. Aerosp. Electron. Syst.* **2024**, *60*, 2071–2081. [CrossRef]
16. Jeon, H.-M.; Kang, T.Y.; Park, J.; Ryoo, C.-K. Attitude control loop design of missile with front lateral thruster using dynamic inversion and greedy algorithm. *J. Inst. Control Robot. Syst.* **2023**, *29*, 620–627. [CrossRef]
17. Lockheed Martin. PAC-3 Missile Segment Enhancement (MSE) Overview. Available online: https://f35.com/content/dam/lockheed-martin/mfc/documents/pac-3/24-09790-iamd-pac-3-mse-partner-ppt--updates_r2.pdf (accessed on 14 March 2026).
18. Lawler, E.L.; Wood, D.E. Branch-and-bound methods: A survey. *Oper. Res.* **1966**, *14*, 699–719. [CrossRef]
19. Tarjan, R. Depth-first search and linear graph algorithms. *SIAM J. Comput.* **1972**, *1*, 146–160. [CrossRef]

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.