The Thirty-Ninth AAAI Conference on Artificial Intelligence (AAAI-25)

DCC: Differentiable Cardinality Constraints for Partial Index Tracking

Wooyeon Jo[1,3], **Hyunsouk Cho**[1,2*]
[1]Department of Artificial Intelligence, Ajou University, Suwon, Republic of Korea
[2]Department of Software and Computer Engineering, Ajou University, Suwon, Republic of Korea
[3]The AI Lab Inc., Seoul, Republic of Korea
{qt5828, hyunsouk}@ajou.ac.kr

Abstract

Index tracking is a popular passive investment strategy aimed at optimizing portfolios, but fully replicating an index can lead to high transaction costs. To address this, partial replication have been proposed. However, the cardinality constraint renders the problem non-convex, non-differentiable, and often NP-hard, leading to the use of heuristic or neural network-based methods, which can be non-interpretable or have NP-hard complexity. To overcome these limitations, We propose a Differentiable Cardinality Constraint (DCC) for index tracking and introduce a floating-point precision-aware method to address implementation issues. We theoretically prove our methods calculate cardinality accurately and enforce actual cardinality with polynomial time complexity. We propose the range of the hyperparameter ensures that our method has no error in real implementations, based on theoretical proof and experiment. Our method applied to mathematical method outperforms baseline methods across various datasets, demonstrating the effectiveness of the identified hyperparameter.

1 Introduction

Index tracking, particularly through full replication, is one of the most widely used strategies in portfolio optimization. This approach constructs a portfolio that mimics a specific market index by including all constituent stocks with corresponding weights. Full replication can be effectively solved as a basic regression problem using mathematical optimization techniques, enabling the efficient and precise portfolio construction. However, this method assigns continuous weights to all stocks in the portfolio, leading to significant transaction costs—a critical challenge in real-world investment scenarios. To mitigate these costs, partial replication has been proposed (Meade and Salkin 1989), (Ertenlice and Kalayci 2018), where only a subset of stocks is assigned weights, reducing the overall number of transactions. Partial replication extends full replication by incorporating a cardinality constraint to limit the number of stocks.

Cardinality constraints, integral to partial replication, exhibit several notable technical challenges (Chang et al. 2000; Pandey and Banerjee 2024): i) Discreteness: Cardinality constraints enforce a limit on the number of selected stocks, resulting in a discrete solution space, unlike the continuous one encountered in full replication. ii) Combinatorial Complexity: These constraints give rise to a combinatorial optimization problem, where all possible combinations must be considered. Their independent and non-continuous nature complicates to reformulate the problem into a form that can be solved using traditional mathematical optimization techniques, such as those requiring linearity, convexity, or differentiability. iii) Computational Complexity: Finding a solution that satisfies the cardinality constraint is classified as an NP-hard problem, characterized by high computational complexity, making it difficult to identify efficient solutions. Due to these inherent characteristics, traditional mathematical optimization approaches, which were effective for solving full replication problems, struggle with partial replication. Consequently, heuristic methods (Beasley, Meade, and Chang 2003; Wu, Kwon, and Costa 2017; Erwin and Engelbrecht 2023; Kabbani 2022; Zheng et al. 2020) have been proposed to address partial replication. However, these heuristic approaches have significant drawbacks, including the non-interpretability of some solution processes and the persistence of high complexity.

To overcome these limitations, it would be advantageous to transform cardinality constraints into a form that can be tackled using mathematical optimization techniques. Thus, we propose the Differentiable Cardinality Constraint (**DCC**), which is not only adaptable to mathematical optimization techniques but also ensures the enforcement of actual cardinality constraints. In summary, our contributions are as follows:

1. We propose **DCC**, applicable to any optimization algorithm handling differentiable constraints, particularly using the Lagrangian multiplier method for partial replication.
2. To address implementation challenges, we introduce a floating-point precision-aware variant, $\mathbf{DCC}_{fpp}$, ensuring accurate enforcement of cardinality constraints.
3. We establish conditions for the constant a in $\mathbf{DCC}_{fpp}$, providing accurate cardinality calculations and constraint enforcement.
4. We validate $\mathbf{DCC}_{fpp}$'s performance in partial replication using the SLSQP method, demonstrating improved re-

*Corresponding Author

sults on real market data and yielding more precise, interpretable solutions within polynomial time.

2 Related Works

Full replication is a passive investment strategy in portfolio optimization, where objective is to minimize the tracking error between a target index and the portfolio index, which can be formulated as a constrained regression problem. This problem can be efficiently solved using mathematical optimization techniques. Specifically, the constraints in full replication include the sum-to-one constraint, where the sum of portfolio weights equals one, and the non-negativity constraint, ensuring that each weight is non-negative. Both constraints are linear, making Quadratic Programming (QP) an effective method for efficiently solving full replication problems, as demonstrated in various studies (Jobson and Korkie 1980; Fabozzi, Markowitz, and Gupta 2011; Boyd and Vandenberghe 2004). Furthermore, since these constraints can also be expressed in differentiable forms, full replication can be solved using Lagrangian multipliers (Shaw, Liu, and Kopman 2008; Bertsekas 2014). These mathematical optimization techniques are easily implemented using libraries such as CVXPY (Diamond and Boyd 2016) or SciPy (Virtanen et al. 2020), which efficiently find precise solutions. Since these methods follow well-established mathematical procedures, the resulting portfolio solutions are interpretable, as the clear objective functions and explicit constraints make it easy to understand how each decision impacts the final outcome. However, the cardinality constraint is neither linear nor differentiable, making it challenging to solve using mathematical optimization methods.

To address this problem, heuristic approaches have been employed to address partial replication problems. Heuristic methods such as search algorithms (Kabbani 2022), which iteratively explore different combinations of stocks to identify those that optimize the portfolio, and meta-heuristic approaches, including evolutionary algorithms (Beasley, Meade, and Chang 2003; Erwin and Engelbrecht 2023), have been used. Additionally, clustering methods (Wu, Kwon, and Costa 2017) have also been employed to select optimal subsets of stocks, effectively reducing the portfolio size while attempting to maintain tracking accuracy. Despite the practical utility of these heuristic methods, they come with inherent limitations. The approximate nature of heuristic solutions means they may find suboptimal solution, and the large search space involved in these problems introduces significant computational complexity. Thus, recently, (Zheng et al. 2020) have proposed the use of neural network-based approaches for partial replication, employing reparameterization techniques to transform unconstrained parameters into forms that satisfy the cardinality constraint. However, these neural network approaches often function as black-box models, obscuring the interpretability of the solutions and the intermediate steps involved.

The limitations of heuristic and neural network approaches underscore the need for a mathematical optimization approach to solve the partial replication problem efficiently. Traditional methods of handling the cardinality constraint, such as iteratively applying full replication and selecting the top K stocks or excluding $N - K$ stocks with the lowest weights, can satisfy the constraint but at the cost of increased complexity and inefficiency. This approach is straightforward but often yields suboptimal results due to its iterative nature and lack of optimization techniques. Unlike these previous methods, we tackle the partial replication problem by proposing the Differentiable Cardinality Constraint (**DCC**), which transforms the cardinality constraint into a differentiable form. This innovation allows it to be directly integrated into mathematical optimization frameworks, enabling efficient and precise resolution of the problem in polynomial time, while still producing interpretable solutions.

3 Preliminaries

Before introducing our Differentiable Cardinality Constraint (**DCC**), it is essential to formally define the index tracking and the cardinality constraint associated with it.

3.1 Full Replication

Traditional index tracking (full replication) involves constructing a portfolio to minimize the difference between the market index and the portfolio index, i.e. tracking error. Minimizing the tracking error is a straightforward regression problem when dealing with N stocks over a duration D. The objective function is $\min \|X\mathbf{w} - \mathbf{y}\|_2^2$ where $X \in \mathbb{R}^{D\times N}$ is the daily return of stocks and $\mathbf{w} \in [0,1]^N$ such that $\mathbf{w} = [w_1 w_2 \dots w_N]^T$ is a weight vector of portfolio. w_i is the weight of i-th stock and $\mathbf{y} \in \mathbb{R}^D$ is the daily target index.

Moreover, the portfolio must satisfy straightforward constraints: each stock should have a non-negative weight, and the sum of all weights must equal one. Then, we can define the full replication problem as:

$$\begin{aligned} \min_{\mathbf{w}} \quad & \|X\mathbf{w} - \mathbf{y}\|_2^2 \\ \text{subject to:} \quad & w_i \geq 0 \ \forall i, \quad \sum_{i=1}^{N} w_i = 1 \end{aligned} \tag{1}$$

However, full replication assigns continuous weights to all stocks, which leads to high transaction costs. Therefore, a cardinality constraint, which limits the number of stocks in the portfolio, is necessary to mitigate these costs.

3.2 Partial Replication

The partial replication ensures that the portfolio's cardinality, calculated through a specific function, does not exceed a given value K. To enforce this constraint, we first define the function that calculates the portfolio's cardinality. Let w_i represent the weight of the i-th stock in the portfolio. Then cardinality function is defined using a binary function, $b(w_i)$, that assigns a value of 0 if a portfolio weight is zero, and 1 if the weight is greater than zero. By summing the binary function values across all weights in the portfolio, we can calculate the portfolio's cardinality. Partial replication extends the full replication approach by incorporating a

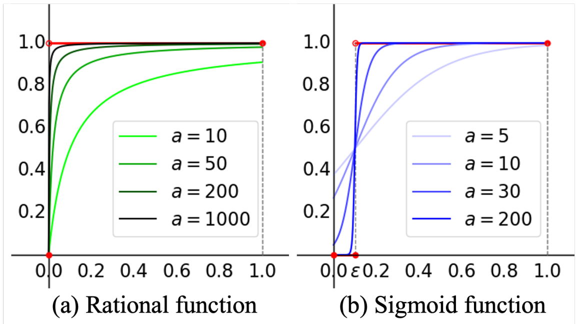


Figure 1: (a) red: binary function, $b(w_i)$. green lines: rational approximated function of $b(w_i)$. (b) red: binary function with cutoff threshold, $b_{fpp}(w_i)$. blue lines: sigmoid approximated function of $b_{fpp}(w_i)$.

cardinality constraint. Using the cardinality function, partial replication can be expressed as follows:

$$\min_{\mathbf{w}} \quad \|X\mathbf{w} - \mathbf{y}\|_2^2$$
$$\text{subject to:} \quad w_i \geq 0 \ \forall i, \quad \sum_{i=1}^{N} w_i = 1 \tag{2}$$
$$C(\mathbf{w}) = \sum_{i=1}^{N} b(w_i) \leq K, \ \ b(w_i) = \begin{cases} 0, & (w_i = 0) \\ 1, & (w_i > 0) \end{cases}$$

where $C(\mathbf{w})$ is the function for calculating cardinality of the portfolio weight $\mathbf{w}$ and K is integer such that $K < N$.

To solve the partial replication problem as a mathematical optimization problem, it is essential to express the cardinality constraint in a form that can be handled by mathematical optimization techniques, particularly in a differentiable manner. Thus we introduct the Differentiable Cardinality Constraint (**DCC**) in Section 4. And the **DCC** must satisfy the following properties with differentiability:

- **Accuracy**: The cardinality function, $C(\mathbf{w})$, in the **DCC** should convert each weight value into the corresponding integer value.
- **Assurance**: The **DCC** must guarantee the limit of the number of selected stocks (K).

4 Differentiable Cardinality Constraint

4.1 Rational Function Approach

Defining the cardinality constraint requires a function that calculates the portfolio's cardinality, $C(\mathbf{w})$. This function can be expressed as the summation of a binary function b. However, as illustrated in Figure 1 (a) (red), this binary function is discontinuous and non-differentiable, rendering $C(\mathbf{w})$, and cardinality constraints are non-differentiable as well. To address this, we approximate the binary function with a differentiable alternative, allowing $C(\mathbf{w})$ and the cardinality constraint to be expressed in a differentiable form under two properties in Preliminaries (Will be discussed in Section 4.2). We utilize the following rational function to approximate the binary function:

$$\tilde{b}(w_i) = 1 - \frac{1}{a \cdot w_i + 1}, \quad a : \text{constant} \tag{3}$$

The graph of the $\tilde{b}(w_i)$ function is shown in Figure 1 (a). This rational function $\tilde{b}(w_i)$ is differentiable for all weight in $[0, 1]$, and it passes through the origin and approaches $\tilde{b} = 1$ as an asymptote, so that it takes the value of 0 when w_i is 0 and approaches 1 for w_1 greater than 0. Here, the constant a can be arbitrarily chosen, and increasing the value of a allows $\tilde{b}(w_i)$ to approximate binary function $b(w_i)$ more closely (See Figure 1 (a) (green lines)). Therefore, selecting a very large value for a is advantageous. Furthermore, the value of a in the approximation function remains independent of the portfolio weight or the number of stocks, thus incurring no additional computational cost or execution time as a increases.

Using $\tilde{b}(w_i)$, the function for calculating cardinality of a portfolio can be approximated. Since $\widetilde{C}(\mathbf{w})$ is composed of differentiable functions of each variable w_i, it is also a differentiable Nth-order function. Therefore, we can get the Differentiable Cardinality Constraint (**DCC**) using $\widetilde{C}(\mathbf{w})$:

$$\widetilde{C}(\mathbf{w}) = \sum_{i=1}^{N} \tilde{b}(w_i) = \sum_{i=1}^{N} \left(1 - \frac{1}{a * w_i + 1}\right) \leq K \tag{4}$$

However, when selecting stocks, portfolio typically sets a weight cutoff threshold. This means that instead of strictly counting weights as 0 when they are exactly zero, the binary function should count a weight as 0 if it is below the cutoff threshold, and as 1 if it is above the threshold. This adjustment accounts for floating-point precision and requires a new binary function that incorporates the cutoff threshold.

4.2 Sigmoid Function Approach (DCC$_{fpp}$)

Cardinality Constraint with Cutoff Threshold Consideration As discussed, due to the floating-point precision issues from cutoff threshold, we redefine the cardinality constraint considering the cutoff threshold of portfolio weights like this:

$$C_{fpp}(\mathbf{w}) \ = \ \sum_{i=1}^{N} b_{fpp}(w_i) \leq K, \tag{5}$$
$$\text{where } b_{fpp}(w_i) = \begin{cases} 0, & \text{if } 0 \leq w_i < \epsilon \\ 1, & \text{if } w_i \geq \epsilon \end{cases}$$

The graph of the redefined binary function is shown in Figure 1 (b) (red). Here, ϵ represents a small cutoff threshold of portfolio weights. Therefore, the redefined cardinality function means that if a weight is less than ϵ, it is counted as zero, and if it is greater than ϵ, it is counted as one. The redefined binary function b_{fpp} from the C_{fpp} remains a non-differentiability. We approximate this again to make it differentiable. However, we can no longer approximate the binary function using a rational function as before. Instead, we transform the sigmoid function to preserve the meaning of the $b_{fpp}(w_i)$ and make it differentiable as follows:

$$\tilde{b}_{fpp}(w_i) = \frac{1}{1 + e^{-a(w_i - \epsilon)}}, \quad a : \text{constant} \tag{6}$$

See Figure 1 (b) (blue lines). As shown in the graph, a larger value of a results in a closer approximation to $b_{fpp}(w_i)$. Since a is a simple constant (for the same reasons as before), choosing a large a does not affect the problem's complexity or execution time. This approximated binary function is differentiable and has an inflection point at $w_i = \epsilon$. Additionally, when a is set sufficiently large, the function has an asymptote at $\tilde{b}_{fpp} = 1$ for weights greater than ϵ and an asymptote at $\tilde{b}_{fpp} = 0$ for weights less than ϵ. Although $\tilde{b}_{fpp}(w_i)$ is 0.5 when $w_i = \epsilon$, the floating-point precision issue means that weights are rarely exactly ϵ. Even if they are, the cardinality constraint is still ensured. This will be discussed further in the next section. To summarize, if a weight is greater than ϵ, $\tilde{b}_{fpp}(w_i)$ is close to 1; if it is less than ϵ, $\tilde{b}_{fpp}(w_i)$ is close to 0.

Similarly, the approximated cardinality function can be defined using the approximated binary function. Since $\widetilde{C}_{fpp}(\mathbf{w})$ is an N-th degree function composed of differentiable terms with respect to each variable w_i, the approximated cardinality function is also differentiable. Thus Differentiable Cardinality Constraint for floating-point precision (**DCC**$_{fpp}$) can be written as follows:

$$\widetilde{C}_{fpp}(\mathbf{w}) = \sum_{i=1}^{N} \tilde{b}_{fpp}(w_i) = \sum_{i=1}^{N} \frac{1}{1 + e^{-a(w_i - \epsilon)}} \leq K \quad (7)$$

Conditions for Accurate Cardinality Calculation In the Preliminaries, one of the key properties that the **DCC**$_{fpp}$ must satisfy is the accurate calculation of the portfolio's cardinality. Achieving this accurate calculation relies on the proper definition of the cardinality function, which, in turn, depends on the binary function used within it. The ability of the cardinality function to accurately reflect the true cardinality is heavily influenced by the value of the constant a used in defining the binary function. Therefore, we establish conditions for the constant a that ensure the cardinality function correctly computes the portfolio's cardinality.

If the value of a is too small, it may count values much larger than zero even when the weight is zero, or conversely, it may fail to count exactly 1 when the weight is 1. In the first case, this could lead to a situation where cardinality is calculated for all weights, regardless of whether they actually contribute to the portfolio. Therefore, to ensure accurate cardinality calculation, the value of a must at least be set such that it counts 0 when the weight is zero and confidently counts 1 when the weight is 1. However, since the $\tilde{b}_{fpp}$ does not exactly take the values of 0 and 1 but instead approaches $b = 0$ and $b = 1$ asymptotically, we consider a bounded condition using the same cutoff threshold value ϵ as mentioned in Section 4.2. We establish the following minimum conditions:

- $C_0 : w_i = 0 \; \forall i \in \{1, \cdots, N\} \Rightarrow \sum_{i=1}^{N} w_i \leq \epsilon$
- $C_1 : w_i = 1 \; \forall i \in \{1, \cdots, N\} \Rightarrow \sum_{i=1}^{N} w_i \geq N - \epsilon$

Conditions for Assurance of the DCC$_{fpp}$ Similarly, we must ensure the second property of **DCC**$_{fpp}$, Assurance. This can be confirmed by verifying that satisfying **DCC**$_{fpp}$ always guarantees the actual cardinality constraints. To prove that our **DCC**$_{fpp}$ ensures the actual cardinality constraint, we need to show the following:

$$\text{If } \sum_{i=1}^{N} \tilde{b}_{fpp}(w_i) \leq K, \quad \text{then } \sum_{i=1}^{N} b_{fpp}(w_i) \leq K$$

Then we present the following theorem:

Theorem 1. *The Assurance of the cardinality constraint*

$$\textit{If } N \cdot err < 1 \quad \textit{and} \quad \sum_{i=1}^{N} \tilde{b}_{fpp}(w_i) \leq K,$$

$$\textit{then } \sum_{i=1}^{N} b_{fpp}(w_i) \leq K$$

N is the number of stocks, e is constant in Lemma 1 (We present proofs and supporting lemmas for Theorem[1].), and K is integer such that $K < N$.

Since N is a fixed value and err depends on the constant a in Eq. 6, if we choose a such that $err < \frac{1}{N}$, then our **DCC**$_{fpp}$ will always ensure the cardinality constraint. Through Theorem 1, we have proven that our **DCC**$_{fpp}$ guarantees the cardinality constraint. In other words, our proposed **DCC**$_{fpp}$ can effectively solve cardinality problem by applying to some optimization algorithms without calculating l_0-norm. This theorem leads to the derivation of last condition for the hyperparameter a that ensure the accurate enforcement of the cardinality constraint.

- $C_2 : err \leq \frac{1}{N}$

Complex Analysis We applied our proposed **DCC**$_{fpp}$ to Sequential Least Squares Quadratic Programming (SLSQP), a mathematical optimization technique using the Lagrangian multiplier method. The time complexity analysis indicates that the addition of **DCC**$_{fpp}$ to SLSQP maintains the algorithm's polynomial time complexity[1].

5 Experiments

In this section, we validate the proposed **DCC**$_{fpp}$ with various dataset in three aspects: 1) We compare index tracking errors to assess the performance of partial replication. 2) We measure the performance of the generated portfolio using commonly used metrics, 3) we compare the runtime of methods to highlight their efficiency. We provide implementation and data loading scripts[1].

5.1 Experimental Settings

Data We conduct experiments using the following three market indices:

1. **S&P 100 Index** : The S&P 100, i.e., Standard & Poor's 100, is a highly representative index that focuses on large, blue-chip companies with high liquidity and stability. As the most critical dataset for our experiments, it is often regarded as a more concentrated and definitive representation of the U.S. market's most significant and stable companies.

[1]Code and proof details: https://github.com/qt5828

2. **S&P 500 Index** : The S&P 500 is a broader stock market index that tracks the performance of 500 large companies listed on U.S. stock exchanges. While it provides a comprehensive overview of the U.S. economy, the S&P 100 is considered a more focused and essential subset within this broader index.
3. **KOSPI 100 Index** : The KOSPI 100, by the Korea Exchange (KRX), tracks the top 100 large-cap stocks in the Korean market, offering insights into large-cap stock trends in South Korea.

For our analysis, we utilized data spanning from January 1, 2018, to April 30, 2023. The data was sourced from Yahoo Finance (Aroussi 2019).

Backtesting We apply the backtesting method presented in (Zheng et al. 2020) to evaluate and compare the tracking performance of each baseline model. The backtesting is conducted using a sliding window technique, where the data period is shifted by one day at a time. On rebalancing days, which are specific days set at regular intervals, the portfolio is adjusted by recalculating and applying new asset weights based on the most recent data. On other days, the performance is assessed using the weights fitted on the most recent rebalancing day. In our study, we rebalance the portfolio on a quarterly basis. For each rebalancing, we utlze one year of historical data to obtain the portfolio weights.

Baselines These baselines are chosen to show that our approach can achieve performance comparable to the state-of-the-art (SOTA) methods.

1. **Stochastic Neural Network-based Model (SNN)** : The SNN model is a state-of-the-art for solving partial replication, known for returning a portfolio with high tracking performance in a short time. During model training, reparametrization is used to express parameters with constraints as unconstrained parameters. It performs well but lacks interpretability due to being a black-box model.
2. **Forward Selection** : The Forward Selection approach satisfies the cardinality constraint by performing $K + 1$ full replications. Each iteration, the model select the highest weight from the remaining stocks with Sequential Leasts Squares Quadratic Programming (SLSQP) algorithm. This process is repeated K times to select a total of K stocks. SLSQP is performed on the selected K stocks to obtain a portfolio that meets the cardinality constraint.
3. **Backward Selection** : In contrast to forward selection, the Backward selection approach performs full replication and iteratively excludes the stock with the smallest weight. This process is repeated until K stocks remain. After $N - K + 1$ iterations, a portfolio that satisfies the cardinality constraint is obtained.

These baseline models are evaluated to demonstrate the efficacy and efficiency of our proposed method in achieving competitive tracking performance while meeting the necessary constraints.

	K=20	K=25	K=30
forward	8.9069	8.2575	7.9333
backward	7.7373	7.5871	8.6562
SNN	5.8007	4.0457	4.9006
$\mathbf{DCC}_{fpp}$	**3.9155**	**3.5385**	**2.3922**

Table 1: Mean Absolute Error (MAE) between the tracking index and the target index

Ours For the fair comparison, we utilize SLSQP with $\mathbf{DCC}_{fpp}$ instead of **DCC** to resolve precision issues, ensuring accurate cardinality enforcement and efficient tracking.

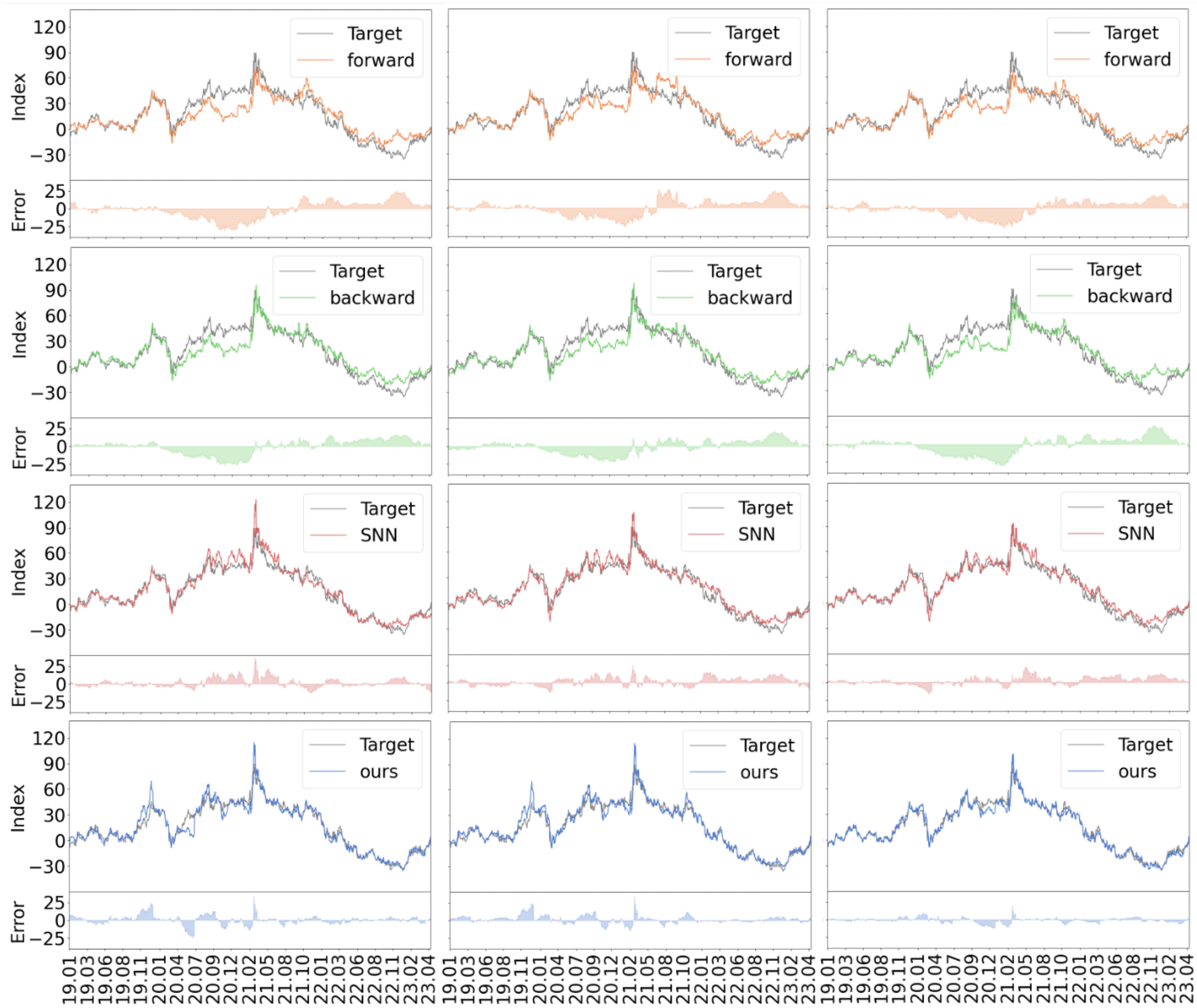
5.2 Index Tracking with Cardinality Constraint

We measure the error between target index and tracking index of partial replication. To evaluate the performance of index tracking. We first fit the portfolio weights on each rebalancing day (3-month) and calculate the tracking index by taking the weighted sum of the returns of each stock. We then plot this tracking index alongside the target index values to visually assess how well each method tracks the S&P 100 index. As illustrated in Figure 2 and summarized in Table 1, our $\mathbf{DCC}_{fpp}$ outperforms the baselines by accurately adhering to the cardinality constraint through a rigorous mathematical procedure.

Our SLSQP with $\mathbf{DCC}_{fpp}$ outperforms the baselines in tracking performance by precisely adhering to the cardinality constraint. Forward and backward selection methods perform poorly because they separate portfolio fitting from asset selection, often leading to suboptimal solutions regardless of the value of K. In contrast, SNN and our method integrate selection and fitting simultaneously, resulting in superior tracking performance. As K decreases, our method effectively reduces the number of selected assets, maintaining strong performance while naturally achieving a slight increase in error, which is expected in scenarios with fewer assets.

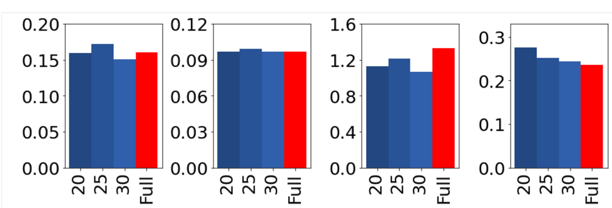
To assess the effectiveness of the portfolios generated through partial replication using our method, we evaluate them using commonly used metrics: cumulative return, volatility, Sharpe ratio, and maximum drawdown (MDD). These four evaluation metrics are aggregated as the averages of the values obtained from all portfolios during the backtesting period. Our method demonstrates performance comparable to that of the full replication (See Figure 3). Despite the cardinality constraints, our portfolio consistently maintains a sharpe-ratio above 1, indicating that it provides favorable returns relative to its risk. Actually, the cumulative return is comparable to full replication, and the volatility shows minimal difference.

To evaluate the robustness of the methods, we also conducted experiments about the KOSPI 100 and S&P 500 indices. The cardinality constraint was set to $K = 25$ and $K = 40$, respectively, and the tracking results are shown in Figure 4. Our SLSQP with $\mathbf{DCC}_{fpp}$ effectively tracks the KOSPI 100 index despite its different distribution compared



(a) K=20 (b) K=25 (c) K=30

Figure 2: Comparison of the tracking indices (orange, green, red, blue) with the S&P 100 target index (grey) over time. Each row indicates SLSQP with forward selection, SLSQP with backward selection, SNN, and SLSQP with $\mathbf{DCC}_{fpp}$, respectively. Additionally, the graph below shows the absolute error between the tracking index and the target index (full cardainlity).



(a) Cumulative Return (b) Volatility (c) Sharpe-Ratio (d) MDD

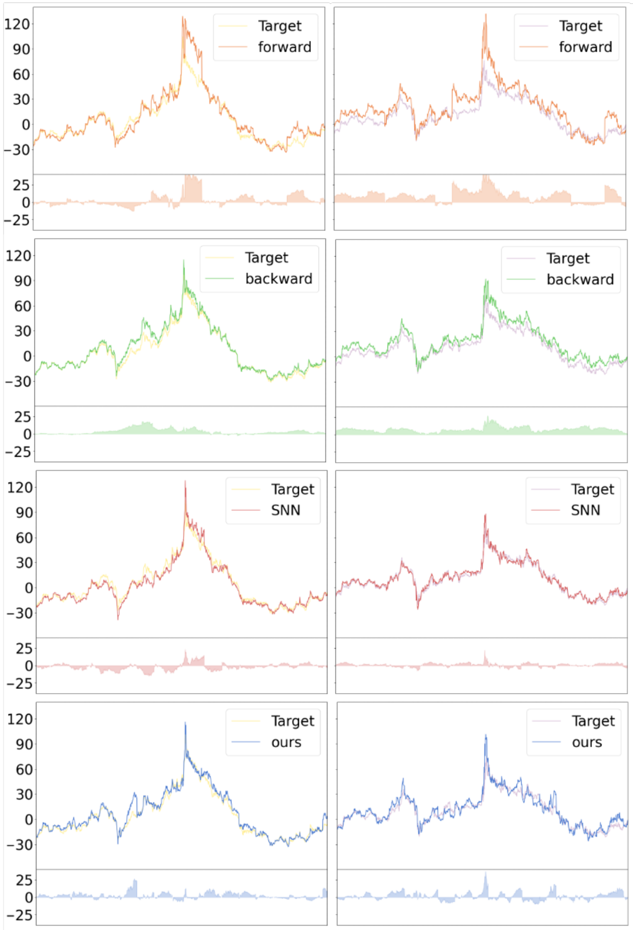
Figure 3: Comparison of secondary evaluations between portfolio of $\mathbf{DCC}_{fpp}$ at different cardinality $K = 20, 25$ and 30 and the portfolio of full replication.

to the S&P 100, and it also demonstrates highly comparable performance on the S&P 500, which has five times the number of stocks as the S&P 100. Our method can efficiently handle index tracking with cardinality constraints across any dataset.

5.3 Efficiency

To illustrate the efficiency of our approach, we compare the runtime taken for index tracking with a cardinality constraint using forward selection and backward selection. The original cardinality constraint is known as an NP-hard problem. Incorporating a cardinality constraint into mathematical optimization algorithms typically results in exponential complexity for the index tracking solution. By applying our proposed $\mathbf{DCC}_{fpp}$, we can find an exact solution that satisfies the existing cardinality constraint within polynomial time.

In Figure 5, the y-axis represents the average runtime to optimize the weights on rebalancing days. As shown in Figure 5, our method (red) is significantly faster than the forward and backward selection. Specifically, forward selection



(a) KOSPI 100 with K=25 (b) S&P 500 with K=40

Figure 4: Comparison of the tracking indices (orange, green, red, blue) on KOSPI 100 and S&P 500. Each rows represents forward, backward, SNN, and $\mathbf{DCC}_{fpp}$, respectively.

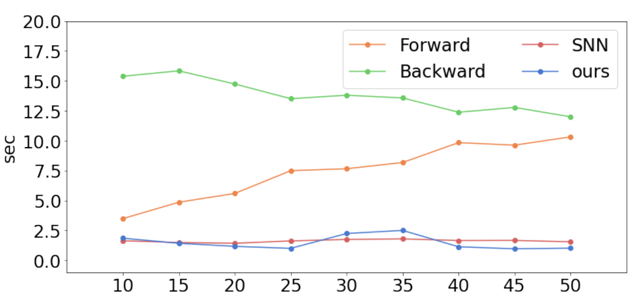


Figure 5: Time comparison of each baseline. Each color indicates method: forward selection (orange), backward selection (green), SNN (red), and $\mathbf{DCC}_{fpp}$ (blue), respectively. The x-axis and y-axis represent the cardinality K and the average time (sec) taken to optimize.

requires $K+1$ iterations of full replication to select the portfolio weights, while backward selection requires $N-K+1$ iterations. Consequently, as K increases, the runtime for forward selection increases, and the runtime for backward selection decreases. Conversely, as K decreases, the runtime for forward selection decreases, and the runtime for backward selection increases. In contrast, our method maintains a consistently low runtime regardless of whether K increases or decreases. This is because our approach directly incorporates the $\mathbf{DCC}_{fpp}$ as a constraint in the mathematical optimization process, eliminating the need for repetitive full replications while still satisfying the cardinality constraint. Also, our method achieves comparable runtime efficiency to the state-of-the-art SNN model, demonstrating its superior efficiency in solving partial replication problems.

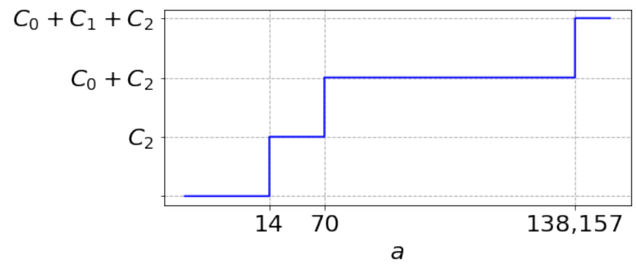


Figure 6: Exploring the range of a (x-axis) satisfying three Conditions, i.e. C_0, C_1 and C_2.

5.4 Hyperparameter Analysis

We also explored the hyperparameter a which determines the satisfiability of the two essential properties (accuracy and assurance) in $\mathbf{DCC}_{fpp}$. To identify the appropriate a value satisfying all three conditions (C_0, C_1 and C_2), we analyzed the status of each condition based on the value of a. As shown in Figure 6, when $a \leq 14$, none of the conditions are met. For $a \geq 14$, C_0 is satisfied, C_2 is satisfied for $a \geq 70$, and finally, C_1 is met when $a \geq 138,157$. a should be set to at least 138,157 to satisfy all conditions in Python's 64-bit floating-point precision, ensuring that our $\mathbf{DCC}_{fpp}$ accurately calculates the portfolio's cardinality and guarantees the enforcement of the cardinality constraint, regardless of the dataset.

6 Conclusion

In this work, we introduced the Differentiable Cardinality Constraint (**DCC**) and its precision-aware variant ($\mathbf{DCC}_{fpp}$) to address the NP-hard problem of partial replication in index tracking. Our method converts the problem into a differentiable form, enabling efficient solutions using mathematical optimization within polynomial time. Experiments show that $\mathbf{DCC}_{fpp}$ achieves comparable performance to full replication while adhering to cardinality constraints, outperforming state-of-the-art heuristic methods in both accuracy and efficiency. The robustness and reduced computational complexity of our approach make it highly applicable in real-world portfolio optimization.

Acknowledgements

This work was supported by the Institute of Information & Communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) (No.2022-0-00680, Abductive inference framework using omni-data for understanding complex causal relations), the National R&D Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Science and ICT (RS-2024-00407282), and the Artificial Intelligence Convergence Innovation Human Resources Development (IITP-2025-RS-2023-00255968) funded by the Korea government (MSIT).

References

Aroussi, R. 2019. yfinance: Download market data from Yahoo! Finance's API. Https://github.com/ranaroussi/yfinance.

Beasley, J. E.; Meade, N.; and Chang, T.-J. 2003. An evolutionary heuristic for the index tracking problem. *European Journal of Operational Research*, 148(3): 621–643.

Bertsekas, D. P. 2014. *Constrained optimization and Lagrange multiplier methods*. Academic press.

Boyd, S.; and Vandenberghe, L. 2004. *Convex optimization*. Cambridge university press.

Chang, T.-J.; Meade, N.; Beasley, J. E.; and Sharaiha, Y. M. 2000. Heuristics for cardinality constrained portfolio optimisation. *Computers & Operations Research*, 27(13): 1271–1302.

Diamond, S.; and Boyd, S. 2016. CVXPY: A Python-embedded modeling language for convex optimization. *Journal of Machine Learning Research*, 17(83): 1–5.

Ertenlice, O.; and Kalayci, C. B. 2018. A survey of swarm intelligence for portfolio optimization: Algorithms and applications. *Swarm and evolutionary computation*, 39: 36–52.

Erwin, K.; and Engelbrecht, A. 2023. Meta-heuristics for portfolio optimization. *Soft Computing*, 27(24): 19045–19073.

Fabozzi, F. J.; Markowitz, H. M.; and Gupta, F. 2011. Portfolio selection. *The Theory and Practice of Investment Management*, 45–78.

Jobson, J. D.; and Korkie, B. 1980. Estimation for Markowitz efficient portfolios. *Journal of the American Statistical Association*, 75(371): 544–554.

Kabbani, T. 2022. Metaheuristic Approach to Solve Portfolio Selection Problem. *arXiv preprint arXiv:2211.17193*.

Meade, N.; and Salkin, G. R. 1989. Index Funds-Construction and Performance Measurement. *The Journal of the Operational Research Society*, 40(10): 871–879.

Pandey, S.; and Banerjee, A. 2024. Cardinality Constraint Non-Uniform Sampling for Maximizing Reconstruction Accuracy of Time-Varying Signals. *IEEE Transactions on Information Theory*, 70(9): 6746–6756.

Shaw, D. X.; Liu, S.; and Kopman, L. 2008. Lagrangian relaxation procedure for cardinality-constrained portfolio optimization. *Optimisation Methods & Software*, 23(3): 411–420.

Virtanen, P.; Gommers, R.; Oliphant, T. E.; Haberland, M.; Reddy, T.; Cournapeau, D.; Burovski, E.; Peterson, P.; Weckesser, W.; Bright, J.; van der Walt, S. J.; Brett, M.; Wilson, J.; Millman, K. J.; Mayorov, N.; Nelson, A. R. J.; Jones, E.; Kern, R.; Larson, E.; Carey, C. J.; Polat, İ.; Feng, Y.; Moore, E. W.; VanderPlas, J.; Laxalde, D.; Perktold, J.; Cimrman, R.; Henriksen, I.; Quintero, E. A.; Harris, C. R.; Archibald, A. M.; Ribeiro, A. H.; Pedregosa, F.; van Mulbregt, P.; and SciPy 1.0 Contributors. 2020. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, 17: 261–272.

Wu, D.; Kwon, R. H.; and Costa, G. 2017. A constrained cluster-based approach for tracking the S&P 500 index. *International Journal of Production Economics*, 193: 222–243.

Zheng, Y.; Chen, B.; Hospedales, T. M.; and Yang, Y. 2020. Index tracking with cardinality constraints: A stochastic neural networks approach. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, 1242–1249.