Optimizing trombone slide positions: a network-based shortest paths approach to musical expression

Joseph Erwin*
University of Southern California

Project Advisor: John Gunnar Carlsson
University of Southern California

Abstract

Selecting optimal slide positions is a crucial challenge in trombone performance that impacts both technical execution and musical expression. This paper models trombone slide positioning as a network-based shortest path problem and presents several objective functions to evaluate different positioning strategies. We compare various cost functions, including minimizing distance to standard positions, finding the global shortest path, a greedy approach, minimizing slide reversals, minimizing impulse changes, and maximizing natural slur opportunities.

Our findings reveal that the shortest path algorithm consistently performs well across different musical pieces, striking a balance between reducing physical effort and maintaining musical expressiveness. The impulse minimization objective is particularly effective for fast, technical passages, while the 'Against the Grain' algorithm excels in slow, melodic passages. The standard positions, though commonly used, are shown to be the least efficient in terms of energy expenditure. Additionally, the greedy algorithm, despite its simplicity, demonstrates robust performance, making it a practical choice for real-time implementation.

These insights not only enhance our understanding of trombone performance optimization but also suggest that adopting advanced computational techniques can significantly improve both the efficiency and expressiveness of musical performance. This study underscores the potential of interdisciplinary approaches to enrich musical execution, offering valuable strategies for trombonists and potentially extending to other musical instruments.

1 Introduction

The strategic planning and coordination of specific movements are crucial elements in musical performance, particularly for musicians playing instruments such as keyboards or strings, where mastering the correct fingering technique is essential for proficiency [5, 4]. These challenges represent a sophisticated combinatorial optimization problem, as seen in the algorithmic approaches applied to fingering decisions using hidden Markov models, variable neighborhood search, and path difference learning [1, 7, 6].

Similar complexities arise in brass instruments, notably the trombone, where the selection of slide positions plays a pivotal role in musical expression. The trombone's unique slide mechanism allows for the production of identical pitches through different positions, facilitated by the overtone series. This flexibility, while beneficial, introduces significant complexity in selecting optimal slide positions for smooth and expressive playing.

Traditional methods of slide position selection often rely on heuristics and may not provide optimal solutions for complex musical passages. The process goes beyond mere technical execution, requiring a deliberate evaluation of how each choice affects the tonal quality. Efficient slide position selection

*Joseph Erwin: University of Southern California, josephdunlaperwin@gmail.com. Project Advisor: John Gunnar Carlsson: University of Southern California, jcarlsso@usc.edu.

not only reduces physical strain for the musician but also enhances the fluidity and expressiveness of the performance.

This paper addresses the challenge of optimizing trombone slide positions to minimize physical effort while maximizing musical expressiveness. We model the problem as a network-based shortest path problem, where each node represents a potential slide position for a note, and each edge represents the transition between positions with an associated cost. The objective is to find the shortest path through the graph that minimizes total slide movement, reduces direction changes, or optimizes factors like impulse changes and natural slur opportunities.

By applying Dijkstra's algorithm and other optimization techniques, this study aims to provide both theoretical insights and practical solutions for enhancing trombone performance. This approach is analogous to the fingering optimization challenges faced by keyboard and string instrument players, and it provides a robust foundation for further exploration of optimization techniques in musical performance.

Notation and Preliminaries

We model the problem of selecting trombone slide positions as finding a shortest path in a directed acyclic graph (DAG), $G = (V, E)$. The graph is constructed as a series of layers, where each layer corresponds to a note in the musical piece.

- **Note Sequence:** A musical piece is represented as a sequence of N notes, indexed by $i \in \{1, 2, ..., N\}$.
- **Layers:** The graph G contains $N + 1$ layers of nodes. The first layer contains a single source node, S, and each subsequent layer i contains nodes representing the possible slide positions for note i.
- **Node (n):** A node $n \in V$ in layer i represents playing note i in a specific slide position. For more complex objectives, a node may also store additional state information, such as the direction of slide travel or velocity.
- **Position (p):** The trombone slide has 7 primary positions. We denote the position for a given node n as p_n. The starting node S is initialized at $p_S = 1$.
- **Edge (e):** An edge $e = (n_{prev}, n_i) \in E$ represents the transition from playing the previous note (at node n_{prev}) to the current note (at node n_i).
- **Cost ($c_{n_{prev} \to n_i}$):** Each edge is assigned a weight, or cost, which is calculated based on the specific objective function being minimized (e.g., distance, impulse change).

2 Methodology

2.1 Dijkstra's Algorithm

To find an optimal pathway through the directed graphs that are the focus of the next section, we have relied on Dijkstra's algorithm, developed by Edsger W. Dijkstra in 1956 [3]. This is a well-known algorithm for finding the shortest paths between nodes in a graph which may represent, for example, road networks, or for our applications, the movement of a trombone slide. This algorithm is used in many applications, such as network routing protocols and geographical mapping [2].

2.1.1 Algorithm Overview

The algorithm maintains a set of nodes whose shortest distance from the source is known and a set of nodes whose shortest distance from the source is yet to be determined. It iteratively selects the node with the smallest known distance, updates the distances of its neighbors, and marks it as visited. This process continues until all nodes have been visited.

Here is a step-by-step outline of Dijkstra's algorithm:

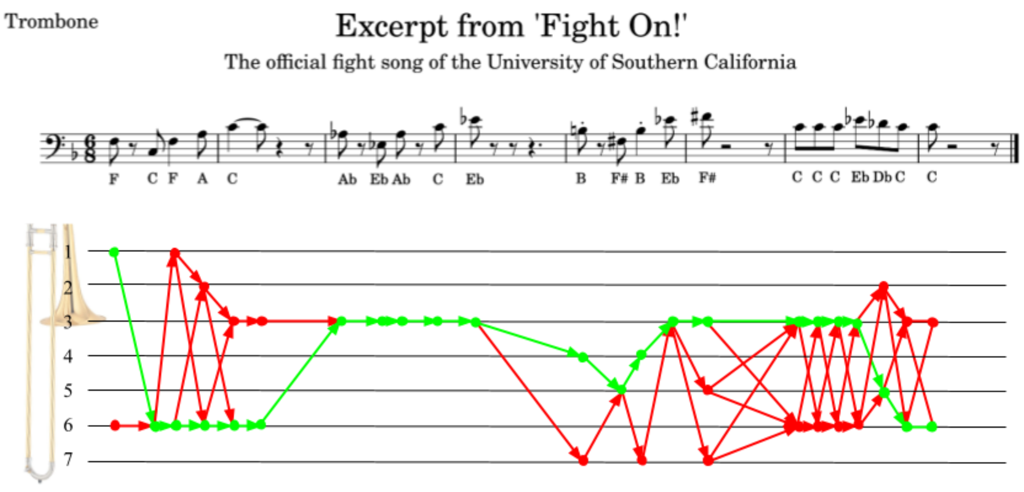
Trombone
Excerpt from 'Fight On!'
The official fight song of the University of Southern California
F C F A C Ab Eb Ab C Eb B F# B Eb F# C C C Eb Db C C
1
2
3
4
5
6
7

Figure 1: Visualization of the shortest path optimization for an excerpt from 'Fight On!'. The horizontal axis represents the sequence of notes in the piece, and the vertical axis represents the seven trombone slide positions. Each point is a node representing a possible position for a given note. The red lines show the entire network of all possible transitions between notes. The green line highlights the single, optimal path found by the shortest path algorithm, which minimizes total slide movement.

1. **Initialization:** Start with the source node, setting its distance to zero and all other nodes' distances to infinity. The source node is added to a priority queue (min-heap).
2. **Iteration:** Extract the node with the smallest distance from the priority queue. For each neighbor of this node, calculate the tentative distance from the source to the neighbor through the current node. If this distance is smaller than the previously known distance, update the distance and add the neighbor to the priority queue.
3. **Completion:** Repeat the iteration step until the priority queue is empty.

2.1.2 Applications in Trombone Slide Position Optimization

In the context of optimizing trombone slide positions, Dijkstra's algorithm can be adapted to find the shortest path in a graph where nodes represent slide positions and edges represent transitions between these positions. By assigning appropriate weights to edges based on physical distance, direction changes, or other criteria, the algorithm helps in determining the sequence of slide positions that best optimize the desired performance metric, an example of which can be seen in Figure 1.

2.2 Descriptions of each Optimizer

In this section we overview each of the optimization objectives, including their mathematical formulations and the pseudocode for the algorithms we have developed.

2.2.1 Pseudocode Variable Descriptions

Across the various algorithms, several common variables and notations are used to construct the state-space graph. Here we define these conventions.

- **General Symbols**
  - **G**: The main directed graph data structure.
  - **S**: The special starting node, representing the initial state before the piece begins.
  - **i**: The index of the current note being processed, iterating from 1 to N.

  - **N**: The total number of notes in the musical piece.

- **Nodes and States**
  - **u, v**: Nodes in the graph G, where a node represents a state. In general, u is a node corresponding to note $i-1$ (the previous state) and v is a node corresponding to note i (the current state).
  - `u.attribute`: Nodes store state information as attributes. For example, `u.position` is the slide position for the state represented by node u. Other attributes like `direction` or `velocity` are used for specific objectives.

- **Sets and Layers**
  - `prev_layer`, `new_layer`: Sets of nodes used to build the graph layer by layer. `prev_layer` holds the nodes for note $i-1$, and `new_layer` is populated with nodes for note i.
  - P_i: The set of all possible slide positions for note i.
  - **p**: A specific slide position from the set P_i.

- **Costs and Edges**
  - `cost`: The weight assigned to an edge from node u to v. This represents the value to be minimized, and its definition changes for each optimization objective.
  - `best_cost`, `best_position`: Temporary variables used in the Greedy algorithm to track the optimal choice at each step.

2.2.2 Standard Positions

This objective attempts to capture the heuristic employed by many novice trombonists. Although many alternate positions exist, the vast majority of players restrict their use of these positions to those initially taught to them, which tend to be the positions closest to first position. We have simply set this objective to attempt to minimize the distance to first position at any given time. This has the following representation:

$$c_{i\to j} = |j-1|.$$

Here, the cost to play a note in position j is simply its distance from first position, regardless of the previous position i.

The process for constructing the graph is detailed in Algorithm 1. The algorithm begins by initializing a graph and a starting node. It then iterates through each note of the musical piece, considering every possible slide position. For each of these possibilities, it creates edges from all the valid positions of the previous note, effectively building the graph layer by layer. The crucial step is where the edge weight (cost) is calculated as the distance of the current position from position 1, in accordance with this objective. New nodes representing a unique note-position combination are added to the graph as they are encountered.

Algorithm 1 Minimize Distance to Common Position

```
Initialize directed graph $G$
Add starting node $S$; $S.position \leftarrow 1$
Set prev_layer $\leftarrow \{S\}$
for each note $i$ from 1 to $N$ do
    Set new_layer $\leftarrow \emptyset$
    for each possible position $p$ in $P_i$ do
        for each node $u$ in prev_layer do
            Compute cost $\leftarrow |p - 1|$
            Let $v$ be the node for position $p$ in the current layer
            if $v$ is not in new_layer then
                Add node $v$ to $G$; $v.position \leftarrow p$
                Add $v$ to new_layer
            end if
            Add edge from $u$ to $v$ with weight cost
        end for
    end for
    Set prev_layer $\leftarrow$ new_layer
end for
return $G$
```

2.2.3 Shortest Path

The shortest path is the simplest and perhaps most obvious objective that cannot be performed in real-time by a musician. In this objective we simply hope to minimize the total distance that the trombone slide moves. The graph construction (see Algorithm 2) is similar to the Standard Positions approach, but the cost function for an edge is modified. Instead of measuring distance to first position, the cost is the absolute distance between the previous note's position and the current note's position:

$$c_{u \to v} = |v.position - u.position|.$$

The problem then becomes finding the shortest path through the resulting graph. Although relatively simple and computationally inexpensive, this solution performs very well even compared to the more complex objectives that follow.

Algorithm 2 Minimize Slide Distance

```
Initialize directed graph $G$
Add starting node $S$; $S.position \leftarrow 1$
Set prev_layer $\leftarrow \{S\}$
for each note $i$ from 1 to $N$ do
    Set new_layer $\leftarrow \emptyset$
    for each possible position $p$ in $P_i$ do
        for each node $u$ in prev_layer do
            Compute cost $\leftarrow |p - u.position|$
            Let $v$ be the node for position $p$ in the current layer
            if $v$ is not in new_layer then
                Add node $v$ to $G$; $v.position \leftarrow p$
                Add $v$ to new_layer
            end if
            Add edge from $u$ to $v$ with weight cost
        end for
    end for
    Set prev_layer $\leftarrow$ new_layer
end for
return $G$
```

2.2.4 Number of Reverses

For this objective, the goal is to minimize the total number of slide direction changes. This implementation does not penalize staying in the same position. To implement this in our model, nodes must store not only the current position but also the direction of travel from the previous node. We define the direction d as a value in $\{-1, 0, 1\}$, representing inward, stationary, and outward motion, respectively. An edge between nodes constitutes a change of direction and is assigned a weight of one, while all other edges have a weight of zero. The cost function is:

$$c_{u\to v} = \begin{cases} 0 & u.direction \times (v.position - u.position) \geq 0, \\ 1 & \text{otherwise.} \end{cases}$$

The full process is shown in Algorithm 3.

Algorithm 3 Minimize Direction Changes

```
1: Initialize directed graph $G$
2: Add starting node $S$; $S.position \leftarrow 1$; $S.direction \leftarrow 0$
3: Set prev_layer ← $\{S\}$
4: for each note $i$ from 1 to $N$ do
5:     Set new_layer ← $\emptyset$
6:     for each possible position $p$ in $P_i$ do
7:         for each node $u$ in prev_layer do
8:             Calculate new_direction ← sign($p - u.position$)
9:             if $u.direction$ × new_direction $< 0$ then
10:                cost ← 1                                ▷ A reversal occurred
11:            else
12:                cost ← 0
13:            end if
14:            Let $v$ be the node for ($p$, new_direction) in the current layer
15:            if $v$ is not in new_layer then
16:                Add node $v$ to $G$; $v.position \leftarrow p$; $v.direction$ ← new_direction
17:                Add $v$ to new_layer
18:            end if
19:            Add edge from $u$ to $v$ with weight cost
20:        end for
21:    end for
22:    Set prev_layer ← new_layer
23: end for
24: return $G$
```

2.2.5 Minimize Impulse

The impulse minimizing objective seeks to minimize the change in momentum of the slide. As in the 'number of reverses' objective, we record additional state information for each node: its position and the velocity required to arrive there. We make a simplifying assumption that the time between notes is constant. The cost for an edge, representing the impulse, is the change in velocity:

$$c_{u\to v} = |(v.position - u.position) - u.velocity|.$$

Because increasing the impulse adds energy to the system and therefore fatigue for the musician, our objective is to minimize this quantity. Algorithm 4 details the graph construction.

Algorithm 4 Minimize Impulse Changes

```
1: Initialize directed graph $G$
2: Add starting node $S$; $S.position \leftarrow 1$; $S.velocity \leftarrow 0$
3: Set prev_layer ← $\{S\}$
4: for each note $i$ from 1 to $N$ do
5:     Set new_layer ← $\emptyset$
6:     for each possible position $p$ in $P_i$ do
7:         for each node $u$ in prev_layer do
8:             Calculate new_velocity ← $p - u.position$
9:             Compute cost ← |new_velocity − $u.velocity$|
10:            Let $v$ be the node for ($p$, new_velocity) in the current layer
11:            if $v$ is not in new_layer then
12:                Add node $v$ to $G$; $v.position \leftarrow p$; $v.velocity$ ← new_velocity
13:                Add $v$ to new_layer
14:            end if
15:            Add edge from $u$ to $v$ with weight cost
16:        end for
17:    end for
18:    Set prev_layer ← new_layer
19: end for
20: return $G$
```

2.2.6 Against the Grain

The 'Against the Grain' objective attempts to maximize the use of natural slurs. A natural slur occurs when extending the slide for a higher note or retracting it for a lower note. In our model (see Algorithm 5), a cost of 0 is assigned to an edge if it produces a natural slur, and a cost of 1 otherwise. This is captured in the following cost function:

$$c_{u \to v} = \begin{cases} 0 & \text{sign}(\text{pitch}(v) - \text{pitch}(u)) \times \text{sign}(v.position - u.position) \geq 0, \\ 1 & \text{otherwise,} \end{cases}$$

where pitch(u) is the pitch of the note associated with node u.

Algorithm 5 Maximize Natural Slurs

```
 1: Initialize directed graph $G$
 2: Add starting node $S$; $S.position \leftarrow 1$; $S.note\_index \leftarrow 0$    ▷ Conceptual note before the first
 3: Set prev_layer $\leftarrow \{S\}$
 4: for each note $i$ from 1 to $N$ do
 5:     Set new_layer $\leftarrow \emptyset$
 6:     for each possible position $p$ in $P_i$ do
 7:         for each node $u$ in prev_layer do
 8:             slide_direction $\leftarrow$ sign($p - u.position$)
 9:             pitch_direction $\leftarrow$ sign(pitch($i$) $-$ pitch($u.note\_index$))
10:             if slide_direction $\times$ pitch_direction ¡ 0 then
11:                 cost $\leftarrow 1$    ▷ Unnatural slur
12:             else
13:                 cost $\leftarrow 0$
14:             end if
15:             Let $v$ be the node for position $p$ in the current layer
16:             if $v$ is not in new_layer then
17:                 Add node $v$ to $G$; $v.position \leftarrow p$; $v.note\_index \leftarrow i$
18:                 Add $v$ to new_layer
19:             end if
20:             Add edge from $u$ to $v$ with weight cost
21:         end for
22:     end for
23:     Set prev_layer $\leftarrow$ new_layer
24: end for
25: return $G$
```

2.2.7 Greedy Algorithm

Similar to the shortest path, the objective for this optimizer is to minimize distance, however here optimization is done locally between notes. For each note, the algorithm greedily selects the slide position closest to the previous note's position. This arrives at a heuristic solution that, albeit sub-optimal, is simple enough to be implemented by a musician manually. As shown in Algorithm 6, this approach builds a single path through the notes and does not require Dijkstra's algorithm for optimization.

Algorithm 6 Greedy Distance Minimization

```
 1: Initialize directed graph $G$
 2: Add starting node $S$; $S.position \leftarrow 1$
 3: Initialize $u \leftarrow S$
 4: for each note $i$ from 1 to $N$ do
 5:     Set best_cost $\leftarrow \infty$
 6:     Set best_position $\leftarrow$ null
 7:     for each possible position $p$ in $P_i$ do
 8:         cost $\leftarrow |p - u.position|$
 9:         if cost ¡ best_cost then
10:             Set best_cost $\leftarrow$ cost
11:             Set best_position $\leftarrow p$
12:         end if
13:     end for
14:     Add new node $v$ to $G$; $v.position \leftarrow$ best_position
15:     Add edge from $u$ to $v$ with weight best_cost
16:     Update $u \leftarrow v$
17: end for
18: return $G$
```

2.2.8 Randomized Baseline

For the sake of comparison, we have included a randomized approach to provide a performance baseline. Unlike the other methods, this algorithm does not build a complete graph for optimization. Instead, it constructs a single, random path, as detailed in Algorithm 7. After initializing a starting node, the algorithm iterates through each note. At each iteration, it randomly selects one of the valid slide positions for the current note, creates a single node for that choice, and adds a single edge connecting the previous node to this new one, before proceeding to the next note. The result is a non-optimal path generated by a series of random local choices.

Algorithm 7 Random Walk Algorithm

```
1: Initialize directed graph $G$
2: Add starting node $S$; $S.position \leftarrow 1$
3: Initialize $u \leftarrow S$
4: for each note $i$ from 1 to $N$ do
5:     Randomly select a position $p$ from $P_i$
6:     Add new node $v$ to $G$; $v.position \leftarrow p$
7:     Compute cost $\leftarrow |v.position - u.position|$
8:     Add edge from $u$ to $v$ with weight cost
9:     Update $u \leftarrow v$
10: end for
11: return $G$
```

3 Use Cases

Each objective has particular strengths and weaknesses. The objective of each is to improve the playing performance of the musician in some way, although different contexts make each objective more or less appropriate.

3.1 Standard Positions

The positions dictated by the standard optimizer have the benefit of being the only sequence that could reasonably be generated in real time by a human player of most any skill level. Simply put, this sequence strictly adheres to the conventional way of playing the trombone. This heuristic is formalized in Algorithm 1, which models this approach by assigning a cost to each note proportional to its distance from first position. Many thousands of musicians have made do with this strategy and the technique is appropriate for any style of playing due in short to the fact that trombone compositions are generally composed with the strengths and limitations of the instrument in mind. Although the trombone is capable of highly virtuosic performance, operating a slide will never be as fast as the valves of a trumpet or keys of a piano.

3.2 Shortest Path

Although the trombone is more limited in the velocity of notes it can produce than some instruments, there is no reason why technology cannot be implemented to address this. The shortest path sequence attempts to reduce the workload for the musician by finding the combination of alternate positions that minimizes the amount of slide movement necessary for the performance of a given piece of music. The process for finding this globally optimal sequence is detailed in Algorithm 2, which builds a complete graph of possibilities where the edge weights are the distances between slide positions. This objective was inspired by the standard use of alternate positions that many advanced trombonists engage in. Although not always appropriate in instances in which perfect tuning is essential, using alternate positions can significantly reduce the workload for a player. As mentioned, this objective may not be appropriate when extreme precision in tuning is necessary, however it functions well for almost every other type of playing. We will subsequently discuss objectives specifically designed for different playing styles, however the shortest path objective performs well in comparison to all of these making it a sort of 'Goldilocks' optimizer.

3.3 Greedy Algorithm

The circumstances in which this objective might be appropriate are largely the same as for the shortest path objective. The greedy algorithm is quite literally a worse version of the shortest path algorithm but with that advantage of being far simpler to solve. Although somewhat of a redundant objective, we include the greedy algorithm to compare the effectiveness of an easy-to-implement algorithm in comparison to increasingly complex ones. As shown in Algorithm 6, the musician need only consider each note and find the alternate position that is closest to the position of the previous note. Although obviously sub-optimal this could be immensely useful for musicians without access to the complex dynamic programming models used for this paper.

3.4 Number of Reverses

In trombone playing one heuristic for minimizing the workload of the given piece is to attempt to identify alternate positions that would reduce the number of direction changes for an excerpt. This is a very common technique, but is often limited to only excerpts of a few notes. Here we find a globally optimized sequence of alternate positions to minimizes the number of direction changes. Algorithm 3 achieves this by encoding the slide's direction of travel into each node's state and assigning a cost only when a reversal occurs. This technique is most well suited for fast paced and virtuosic playing. One of the goals of any accomplished trombonist is to maintain fluid slide movement. This reduces the chance of overshooting a position causing tuning a note attack issues. By minimizing the number of slide reverses, we create a sequence that provides the most fluid slide work possible. Although excellent for fluidity, in considering only the direction of the slide movement between notes, this objective has the notable disadvantage of produces sequences antithetical to minimizing slide movement. These sequences are almost amusing in some sense as they generally involve extending the slide to its furthest position and then retracting to first position. Although highly fluid this style of playing is an unlikely candidate to be adopted by any serious musician, although for comedic brass ensembles this may be a good choice.

3.5 Minimize Impulse

The impulse minimizing objective attempts to address the shortcomings of reducing the number of slide direction changes and minimizing the total distance. Rapidly going from first position to sixth position and then back to first is only a single direction change but incurs a massive expense in distance traveled. Similarly, alternating between third and fourth positions may minimize distance traveled, but will come at the cost of rapid direction changes. Impulse, or change of momentum, was the obvious choice to address these issues because it considers not just the velocity of the slide but also the direction of travel. For the single dimensional case that is the trombone slide, this results in solutions that smooth over rapid changes of direction and speed. Algorithm 4 implements this by storing both position and velocity in each node's state, defining the transition cost as the change in velocity.

Theoretically, this should results in the most ergonomic trombone playing experience. As a result, this objective is appropriate for any style of trombone playing. It is well equipped to handle fast, virtuosic passages, but would also be helpful in standard orchestral or ensemble playing.

3.6 Against the Grain

Against the grain' is a term interchangeably used with natural slur' to refer to the use of the property of the trombone in which moving the slide in opposition to the direction of the note will result in a natural slur, saving the musician the laborious task of tonguing the note. This is a useful property that helps the trombonist feel more at home among other brass instrument players who play instruments with valves, each of which naturally re-articulates when the valve is depressed or released. The Against the Grain objective seeks to use as many natural slurs as possible using this property. As detailed in Algorithm 5, this is achieved by assigning a cost to any transition where slide direction and pitch direction are in opposition, then finding the path that minimizes this cost. Although convenient in general, this objective is most appropriate for legato or smooth and flowing–playing. Using natural slurs here can create a more natural and connected feeling to the music as natural slurs are often less

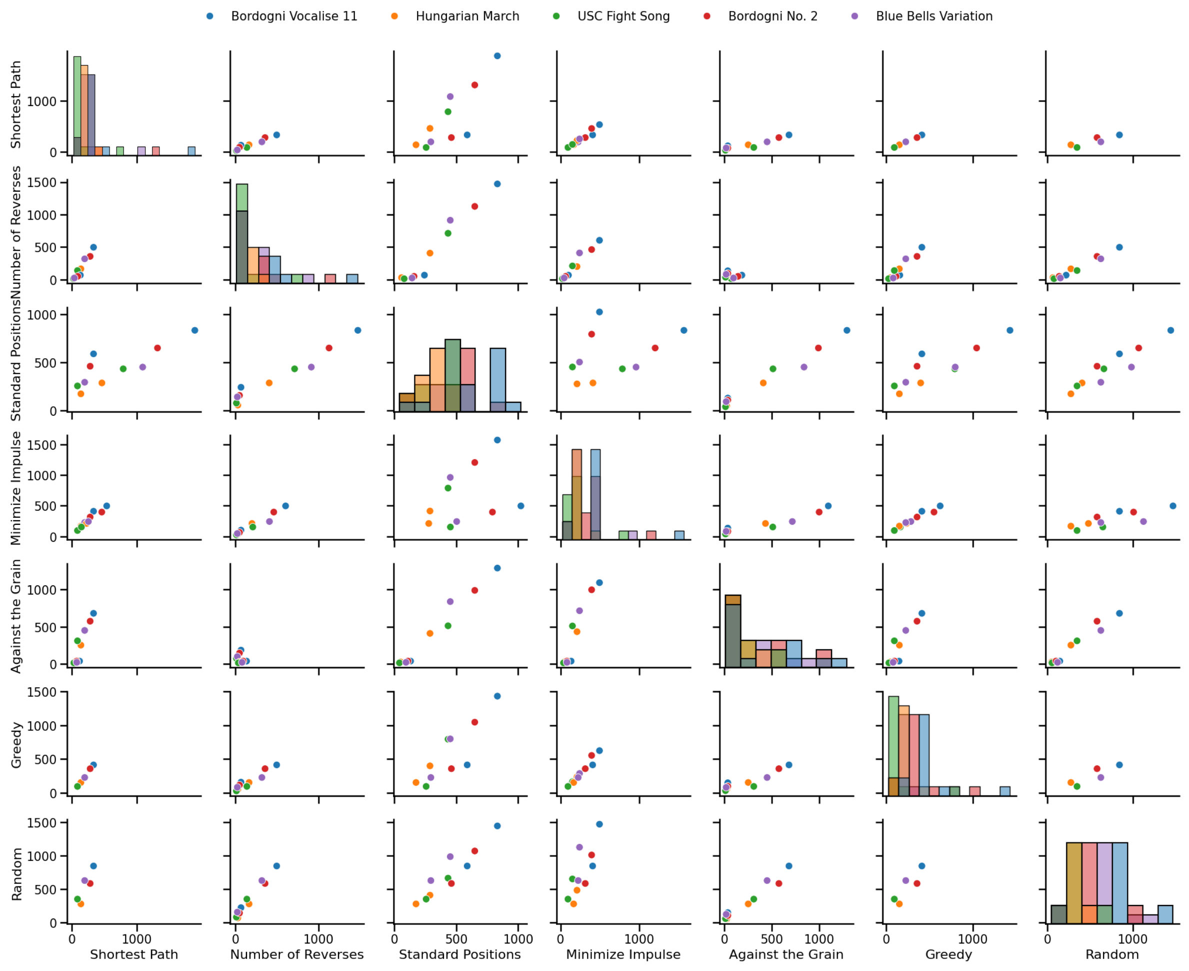

Figure 2: Cross-Objective Pair Plot. The diagonal shows histograms of the fitness scores for each of the five musical pieces. The off-diagonal scatter plots compare the performance of paths generated by one optimizer (x-axis) when scored by another's metric (y-axis).

harsh than tongued re-articulations. Using natural slurs is a common technique for this style of playing already, so this objective is simply designed to double-down on this approach.

4 Performance Comparison

The optimizers were implemented in Python, using the networkx library to construct and solve the graph problems. Because each optimizer is best suited for different types of playing, the performance of each is highly variable. To best understand the trade-offs between the objectives, we have constructed Figure 2 to visually represent these differences. Additionally the percentile rank for each objective pairing for each of the 5 pieces of music being analyzed can be found in Appendix A. On each axis are listed all six objective functions. At the intersection of each grid point is a graph that shows the fitness of each optimal path for the x-axis according to the objective on the y-axis. For example, the optimal path for the Hungarian March according to the shortest path algorithm has a fitness of 128–that is the distance the slide moves is 128 positions. When evaluating the same path using the number of reverses algorithm, the fitness is 150, or this path requires 150 slide direction changes. This plot gives a good impression of how some objectives seem to be almost interchangeable and how others are completely antithetical. Along the diagonal is a histogram of the fitness values for each piece according to the corresponding optimizers on the x- or y-axes.

4.1 Insights

As mentioned, each piece has a different style and therefore is more conducive to different types of playing. This becomes apparent in observing Figure 2. For each piece, the dispersion of the similarly colored points is somewhat indicative of stylistic incompatibility. By looking at these relationships, we can uncover various insights that may be instructive for more efficient trombone playing.

4.2 Inefficiency of Standard Positions

A key insight from the cross-objective analysis is the trade-off offered by the Standard Positions optimizer (Algorithm 1). As shown in the appendix tables, this method, which relies on the most familiar and commonly taught slide positions, rarely performs optimally on any metric related to physical efficiency. When measured against objectives like Shortest Path or Minimize Impulse, the path generated by the Standard Positions optimizer is demonstrably less efficient than paths from more specialized approaches. For instance, in 'Bordogni Vocalise 11', the standard path ranks in the 43rd percentile for total distance traveled (Table A.1), indicating significantly more movement than necessary.

The primary benefit of the standard positions is their ease of use and low cognitive load for the musician, as they require no real-time calculation of alternatives. This benefit should not be discounted, particularly for inexperienced players or in sight-reading scenarios. However, the data confirm that this simplicity comes at the cost of physical efficiency, suggesting that for prepared performance, alternative strategies can yield significant ergonomic improvements.

4.3 The Shortest Path as a Balanced Heuristic

The Shortest Path algorithm (Algorithm 2) was initially envisioned simply to minimize the total distance the slide travels. However, the results indicate it serves as a strong, balanced starting point for optimization. Across various pieces and metrics, it consistently avoids the worst-possible outcomes and often provides a reasonable approximation of more complex objectives. For example, in the 'Blue Bells Variation', the shortest path also scores in the 86th percentile for minimizing impulse, demonstrating a strong positive correlation between these two physical objectives (Table A.5).

This Goldilocks' quality does not mean it is a universal solution. It is often outperformed by specialized objectives; for example, it ranks only modestly on the Slurs (ATG) metric for the Hungarian March' (Table A.2), where the Minimize Impulse objective (Algorithm 4), which encodes more information, scores better. Nonetheless, its performance suggests that simply minimizing total distance is a powerful and effective heuristic that incidentally improves other aspects of playability, making it a valuable tool for general-purpose optimization.

4.4 Robustness of the Greedy Algorithm

Because the greedy algorithm (Algorithm 6) is a heuristic and does not optimize globally, we would expect it to perform somewhat poorly in comparison to the more sophisticated algorithms. Although this is true to some extent, the performance appears to be much better than expected. The algorithm seems to fair well in comparison to all other optimizers with the exception of the standard positions, which we have already established to be poor. Because the greedy algorithm is intended as a proxy for the shortest path algorithm, we see a very close coupling between the two with virtually no dispersion. We have established that the shortest path algorithm is the 'Goldilocks' algorithm so the similar performance of the greedy algorithm to the shortest path imparts the similar level of good performance relative to other objectives. The primary limitation of the greedy algorithm is that it looses considerable efficiency in it's limited planning horizon. The algorithm simply selects the closest viable position which often will result in a needless number of direction changes and large slide movements. The other objectives effectively have a concept of what general area the slide should occupy now to reduce the cost of movement later. Nonetheless, the limited number of positions on the trombone likely contribute to the greedy algorithm exhibiting good performance even compared to the most sophisticated alternatives.

This is particularly notable as this algorithm is entirely possible to implement on-the-fly. All that is needed for a musician to implement this is knowledge of the alternate positions for the notes he will be expected to play. With a little practice, the process of using these alternate positions can become quite trivial and the musician can do so with knowledge that they are playing the trombone in a near optimal way.

4.5 Musical Insights

4.5.1 Melodic Playing

Different pieces of music often require different styles. One purpose in evaluating multiple objective functions is to find ways to play these pieces that are more aligned with their styles. The Bordogni Vocalise 11 also known as the Rochut Number 11 from the Melodious Etudes book requires a melodic style with near constant use of legato. In effect, almost every note needs to be slurred. Most players accomplish this by using their tongues to create the slur, but a natural slur will always be cleaner and more natural sounding. This piece is a prime candidate for the Against the Grain' objective (Algorithm 5). Indeed in analyzing the performance between the standard positions and the 'Against the Grain', the standard positions create 181 note changes without a natural slur. The Against the Grain' algorithm only creates 33. This is clearly a significant improvement that could fundamentally alter the way this piece is performed. One caveat is that by prioritizing only the number of natural slurs, the sequence of positions can be quite strange and often involve large slide movements. Fortunately, musical compositions of this style tend to be slow and prioritize the beauty of the music rather than the technique involved in it's execution.

4.5.2 Fast Playing

By contrast, some pieces emphasize technical execution and require extreme efficiency in transitioning between notes. The Blue Bells of Scotland is a piece that typifies this style. This piece is often performed for no other reason than to demonstrate a supreme level of command over the instrument. The impulse minimizing objective (Algorithm 4) was designed with this piece in mind because it includes a broad range of notes, conventionally played in many positions of the instrument and all at a breakneck tempo. The impulse minimizing objective attempts to reduce rapid direction changes and create long, unbroken runs on the slide. This reduces the work the arm needs to do and allows the player to focus more on tonguing and anticipating the next notes. For this piece, this objective function outperforms all other objectives by a large margin with the exception of the shortest path. The optimal impulse fitness value for the Blue Bells of Scotland is 402 and the shortest path algorithm generates a fitness value of 406. This discrepancy is likely explained by a change in direction when the slide had momentum, but for a vastly more simple algorithm, the shortest path is excellent in this regard. The greedy algorithm also performs well for this purpose, outperforming all other objective functions by a wide margin apart from those already discussed. As discussed the greedy algorithm can be easily implemented mentally (albeit perhaps not for this piece), so it's relatively strong performance is encouraging.

As has been a recurring trend, the optimal sequence of positions often appears strange. This piece is no exception, however, the process of learning this piece is inherently arduous, making it an excellent candidate for optimized alternate positions. Although never attempted, using an algorithm such as this to optimize the positioning could result in even more impressive virtuosic performances. Certainly the level of comfort in using these unconventional positions could negatively impact performance, but as is often the case, unconventional strategies yield unconventional results and sometimes even for the better.

5 Conclusion

In conclusion, this paper has explored the multifaceted optimization problem presented by trombone slide positioning, drawing parallels with challenges faced by other musical instruments. By employing mathematical models and optimization techniques, we have delved into various strategies aimed at balancing physical exertion with musical expressiveness. The analysis reveals that while traditional playing methods prioritize ease and familiarity, they may not always align with the most efficient or expressive playing techniques. Our exploration of different optimization objectives demonstrates that there is no one-size-fits-all solution; rather, the optimal approach may vary depending on the musical context and the specific demands of the piece being performed. In most cases, it is sufficient to rely on the shortest path algorithm as this seems to perform well when compared against all other objectives. For fast, technical passages the impulse objective is superior. For slow, melodic passages the ‘Against the Grain’ algorithm dominates. Finally, in situations in which robust optimization is not available, the greedy algorithm performs very well across all objectives.

The quantitative analysis presented here not only sheds light on the intricacies of trombone performance but also contributes to a broader understanding of optimization in musical execution. By integrating engineering principles with musical artistry, this research offers valuable insights that could enhance performance practices, potentially extending to a wide array of musical instruments. Ultimately, this study underscores the potential of interdisciplinary approaches to enrich our understanding and execution of musical performance, paving the way for innovative strategies that harmonize the technical and expressive facets of playing the trombone and other instruments.

References

[1] Matteo Balliauw, Dorien Herremans, Daniel Palhazi Cuervo, and Kenneth Sörensen. A variable neighborhood search algorithm to generate piano fingerings for polyphonic sheet music. *International Transactions in Operational Research*, 24(3):509–535, 2017.

[2] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms*. MIT Press, 3rd edition, 2009.

[3] Edsger W. Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1(1):269–271, 1959.

[4] Gen Hori and Shigeki Sagayama. Minimax viterbi algorithm for hmm-based guitar fingering decision. In *ISMIR*, pages 448–453, 2016.

[5] Eita Nakamura, Yasuyuki Saito, and Kazuyoshi Yoshii. Statistical learning and estimation of piano fingering. *Information Sciences*, 517:68–85, 2020.

[6] Aleksander Radisavljevic and Peter F Driessen. Path difference learning for guitar fingering problem. In *ICMC*, volume 28. Citeseer, 2004.

[7] Yuichiro Yonebayashi, Hirokazu Kameoka, and Shigeki Sagayama. Automatic decision of piano fingering based on a hidden markov models. In *IJCAI*, volume 7, pages 2915–2921. Citeseer, 2007.

Appendices

A Objective Pairing Percentiles

The following tables show how the optimal path generated by one objective (the row) performs when it is scored using a different objective's metric (the column).

To calculate each value in a column, we first take the seven unique paths generated by our seven optimizers. We score all seven of those paths using the metric for that column (e.g., 'Number of Reverses'). This creates a distribution of seven scores. The number in the table represents the percentile rank of a particular path's score within that distribution.

A rank of 1.0 indicates that a path performed best on that metric compared to all other paths (highlighted in green), while the lowest value indicates the worst performance (highlighted in pink).

A.1 Bordogni Vocalise 11

Table 1: Percentile Ranks for Bordogni Vocalise 11

	Shortest Path	Reverses	Standard	Impulse	Slurs (ATG)	Greedy	Random
Shortest Path	1.000	0.714	0.143	0.857	0.857	1.000	1.000
Number of Reverses	0.571	1.000	0.429	0.714	0.571	0.571	0.571
Standard Positions	0.429	0.143	1.000	0.429	0.714	0.429	0.429
Minimize Impulse	0.857	0.857	0.286	1.000	0.429	0.857	0.857
Against the Grain	0.286	0.429	0.857	0.286	1.000	0.286	0.286
Greedy	0.714	0.571	0.714	0.571	0.143	0.714	0.714
Random	0.143	0.286	0.571	0.143	0.286	0.143	0.143

A.2 Hungarian March

Table 2: Percentile Ranks for Hungarian March

	Shortest Path	Reverses	Standard	Impulse	Slurs (ATG)	Greedy	Random
Shortest Path	1.000	0.571	0.143	0.714	0.714	1.000	1.000
Number of Reverses	0.714	1.000	0.714	1.000	0.143	0.714	0.714
Standard Positions	0.429	0.429	1.000	0.429	0.286	0.429	0.429
Minimize Impulse	0.571	0.857	0.429	0.857	0.857	0.571	0.571
Against the Grain	0.143	0.143	0.571	0.143	1.000	0.143	0.143
Greedy	0.857	0.714	0.857	0.571	0.286	0.857	0.857
Random	0.286	0.286	0.286	0.286	0.286	0.286	0.286

A.3 USC Fight Song

Table 3: Percentile Ranks for USC Fight Song

	Shortest Path	Reverses	Standard	Impulse	Slurs (ATG)	Greedy	Random
Shortest Path	1.000	0.857	0.286	1.000	0.857	1.000	1.000
Number of Reverses	0.571	1.000	0.571	0.571	0.571	0.571	0.571
Standard Positions	0.429	0.143	1.000	0.429	0.286	0.429	0.429
Minimize Impulse	0.714	0.714	0.429	0.857	0.429	0.714	0.714
Against the Grain	0.286	0.286	0.857	0.286	1.000	0.286	0.286
Greedy	0.714	0.571	0.143	0.714	0.714	0.714	0.714
Random	0.143	0.429	0.714	0.143	0.143	0.143	0.143

A.4 Bordogni No. 2

Table 4: Percentile Ranks for Bordogni No. 2

	Shortest Path	Reverses	Standard	Impulse	Slurs (ATG)	Greedy	Random
Shortest Path	1.000	0.714	0.143	0.857	0.857	1.000	1.000
Number of Reverses	0.714	1.000	0.429	0.714	0.571	0.714	0.714
Standard Positions	0.429	0.143	1.000	0.429	0.143	0.429	0.429
Minimize Impulse	0.857	0.857	0.286	1.000	0.714	0.857	0.857
Against the Grain	0.143	0.286	0.857	0.143	1.000	0.143	0.143
Greedy	0.571	0.571	0.571	0.571	0.429	0.571	0.571
Random	0.286	0.429	0.714	0.286	0.286	0.286	0.286

A.5 Blue Bells Variation

Table 5: Percentile Ranks for Blue Bells Variation

	Shortest Path	Reverses	Standard	Impulse	Slurs (ATG)	Greedy	Random
Shortest Path	1.000	0.857	0.143	0.857	0.857	1.000	1.000
Number of Reverses	0.429	1.000	0.571	0.571	0.714	0.429	0.429
Standard Positions	0.571	0.143	1.000	0.429	0.286	0.571	0.571
Minimize Impulse	0.857	0.714	0.429	1.000	0.429	0.857	0.857
Against the Grain	0.286	0.429	0.714	0.286	1.000	0.286	0.286
Greedy	0.714	0.571	0.857	0.714	0.571	0.714	0.714
Random	0.143	0.286	0.286	0.143	0.143	0.143	0.143