Supplementary Material for Submission ID: paper1104
Improved Functional Mappings via Product Preservation

Proof of Theorem 1:
We consider the following problem:

(APPROX):

INPUT: \( N \in \mathbb{N}, K \) "basis" functions \( \varphi_1, \ldots, \varphi_K : \{1, \ldots, N\} \to \mathbb{R} \), "target" function \( f : \{1, \ldots, N\} \to \mathbb{R} \), \( \epsilon > 0 \), a cost \( c \in \mathbb{N} \)

OUTPUT: \( g_1, 1, \ldots, g_1, r_1, \ldots, g_P, 1, \ldots, g_P, r_P \in \{1, \ldots, K\} \), \( \alpha_1, \ldots, \alpha_P \in \mathbb{R} \) such that:

- \( \|f - \sum_{i=1}^P \alpha_i \cdot G_i\|_\infty < \epsilon \), where \( G_i = \prod_{j=1}^{r_i} \varphi_{g_{i,j}} \)
- \( \sum_{i=1}^P (r_i - 1) \leq c \)

and the boolean TRUE, if such a construction exists.
The boolean answer FALSE if no such construction exists.

We want to show this problem \( NP \)-hard. For this, we will make a polynomial reduction from 3 – SAT:

(3-SAT):

INPUT: \( X_1, \ldots, X_Q \) boolean variables, \((T_{1,1} \lor T_{1,2} \lor T_{1,3}) \land \ldots \land (T_{M,1} \lor T_{M,2} \lor T_{M,3})\) a boolean formula, where each \( T_{m,v} \) is some \( X_i \) or some \( \bar{X}_i \) (negation of \( X_i \)).

OUTPUT: The boolean TRUE, a function \( g : \{1, \ldots, Q\} \to \{FALSE, TRUE\} \) such that assigning each variable \( X_q \) to \( g(q) \) satisfies the above boolean formula described by the \( T_{m,j}s \).
The boolean FALSE if no assignment of \( X_q \) to boolean values can satisfy the above boolean formula.

Reduction from (3-SAT) to (APPROX):
We assume that we are given an initial instance of (3-SAT) given with the above notations. We will construct an instance of (APPROX) whose solution will be proven convertible into a solution of the initial problem.

Summary of the proof:

STEP 1:
We will define clusters of variables \( n \in \{1, \ldots, N\} \) over which \( f \) and a cluster of basis functions \( f_k \) for some \( k \in \{1, \ldots, K\} \) will take a non-zero value.

STEP 2:
We will prove that each of the \( Q + M \) pairs of clusters (variables - functions) defined incurs a cost \( \geq 1 \), therefore the total cost is \( \geq Q + M \). In the reduction that we propose, the initial instance of (3-SAT) will have a solution if and only if the total cost of the corresponding (APPROX) that we built is equal to \( Q + M \).

Therefore, we can deduce:
Lemma: any solution (APPROX) that we consider can be supposed to involve only 1 product for each pair of clusters.

STEP 3:
We show the first direction: if the initial instance of (3-SAT) has a solution then we have a solution to our constructed instance of (APPROX).

The reader should then be able to guess from the structure of this solution how we can prove the other direction.

1
STEP 1:

We fix $\epsilon$ very small, for example $\epsilon = \frac{1}{\sqrt{3M+7}}$ will work. For our reduction, we will independently define some basis functions over some $n \in \{1, \ldots, N\}$, specifically designed for encoding the initial instance of ($3$-$\text{SAT}$). Because we want to design these functions over 3 independent set of values taken by $n$, we will change the notation: we fix $N = 7Q + 6M + M^2$ but for convenience of notation, instead of using an index $1 \leq n \leq N$, we will use indices $(1, q, j)$ with $1 \leq q \leq Q$ and $1 \leq j \leq 7$, indices $(2, m, j)$ with $1 \leq m \leq M$ and $1 \leq j \leq 6$ and $(3, m, m')$ with $1 \leq m, m' \leq M$, as in Figure 1. Likewise, we fix $K = 3Q + M^2 + 3M$ but we will use indices $(4, q, j)$ for $1 \leq j \leq 3$, $(5, m, m')$ and $(6, m, j)$ for $1 \leq j \leq 3$, as in Figure 2.

Now we want to define the values taken by the functions $f$ and $\varphi$ over $n \in \{1, \ldots, N\}$ in such a way that:

- $f$, $\varphi(4,q,\cdot)$ are essentially the only functions that may take a non-zero value at $(1,q,\cdot)$, as shown on Figure 3
- $f$, $\varphi(5,m,\cdot)$, $\varphi(6,m,\cdot)$ are (exactly) the only functions that may take a non-zero value at $(2,m,\cdot)$, as shown on Figure 4

In each of the previous cases, we will define the values taken in such a way that, in order to approximate $f$, the only option will be to make a product of some kind.

Before explaining the behavior that our definitions will induce over a potential solution, let’s define the
values taken by each function so that the reader can refer to it (or refer to Figures 3, 4) to follow the analysis:

- **Target function:**
  \[ f((1, q, j)) = 1 \text{ if } j = 1, \quad f((2, m, j)) = 1 \text{ if } j = 1, \quad f(\cdot) = 0 \text{ in all other cases.} \]

- **Basis functions** \( \varphi_{(4,q,\cdot)}: \varphi_{(4,q,1)}((1, q, j)) = 1 \text{ if } j \in \{1, 2, 4, 6\}, \varphi_{(4,q,1)}(\cdot) = 0 \text{ everywhere else}. \)
  \( \varphi_{(4,q,2)}((1, q, j)) = 1 \text{ if } j \in \{1, 3, 5\}, -1 \text{ if } j = 4, 0 \text{ for other cases}. \)
  \( \varphi_{(4,q,3)}((1, q, j)) = 1 \text{ if } j \in \{1, 3, 7\}, -1 \text{ if } j = 6, 0 \text{ for other cases}. \)

- **Basis functions** \( \varphi_{(5,m,m')}:\)
  \( \varphi_{(5,m,m')}((2, m, j)) = 1 \text{ if } j \in \{1, 2\}, \varphi_{(5,m,m')}((3, m, m')) = M, \varphi_{(5,m,m')}((1, q, j)) = \frac{1}{m} \text{ if } \exists v: (T_{m,v} = X_q \text{ & } j = 4) \text{ or } (T_{m,v} = X_q \text{ & } j = 6), \varphi_{(5,m,m')}(\cdot) = 0 \text{ for other cases}. \)

- **Basis functions** \( \varphi_{(6,m,\cdot)}:\)
  \( \varphi_{(6,m,j)}((2, m, j')) = 1 \text{ if } j' \in \{1, 3, 3 + j\}, \varphi_{(6,m,j)}((1, q, j')) = 1 \text{ if } (T_{m,j} = X_q \text{ & } j' = 4) \text{ or } (T_{m,j} = X_q \text{ & } j' = 6), 0 \text{ for other cases}. \)

**STEP 2:**

On figure 3 we can see the values taken by \( f, \varphi_{(4,q,1)}, \varphi_{(4,q,2)} \text{ and } \varphi_{(4,q,3)} \) over \((1, q, 1), (1, q, 2), \ldots, (1, q, 7). \)**

**Key step:** Because no other function will take a non-zero value at \((1, q, j) \neq 4, 6\), we will be able to deduce that in order to approximate \( f \) there will be no other option than making at least 1 product. Intuitively we can see it because if we try to approximate the value \( f((1, q, 1)) = 1 \), using \( \varphi_{(4,q,1)} \) (resp. \( \varphi_{(4,q,2)}, \varphi_{(4,q,3)} \)) would poorly approximate \( f \) at \((1, q, 2) \) (resp. \((1, q, 5), (1, q, 7). \))

On figure 4 we can see the values taken by \( f, \varphi_{(5,m,m')} \text{ and } \varphi_{(6,m,j')} \) over \((2, m, \cdot). \)**

**Key step:** As before, because no other function will take a non-zero value at \((2, m, \cdot), \) we can deduce that in order to approximate \( f \) there will be no other option than making at least 1 product.

We formally prove these two key steps below by evaluating some functions at values \( 1 \leq n \leq N. \)

For convenience of notation, we will write \( \alpha(G_p) \) instead of \( \alpha_p, \) and we may also use this notation for functions \( G_p \) not built by the solution. In our notation, this will be equivalent to \( \alpha(G_p) = 0. \)

Let \( H(n) \) denote the property: \(|f(n) - \sum_{i=1}^{P} \alpha_i \cdot G_i(n)| < \epsilon \) obtained from the evaluation of \( \|f - \sum_{i=1}^{P} \alpha_i \cdot G_i\|_{\infty} < \epsilon \) at \( n. \)

Evaluation at \((1, q, \cdot): \)

- \( H((1, q, 2)) \Rightarrow |\alpha(\varphi_{(4,q,1)})| < \epsilon \)
\[ H((1,q,5)) \Rightarrow |\alpha(\varphi(4,q,2))| < \epsilon \]
\[ H((1,q,7)) \Rightarrow |\alpha(\varphi(4,q,3))| < \epsilon \]
\[ H((1,q,1)) \Rightarrow \text{Since } \epsilon \leq \frac{1}{5}, \text{ at least one product is made among the functions } \varphi(4,q,1), \varphi(4,q,2), \varphi(4,q,3) \]

Evaluation at (2, m,) and (3, m, m'):
\[ H((3,m,m')) \Rightarrow |\alpha(\varphi(5,m,m'))| < \frac{\epsilon}{M} \]
\[ H((2,m,3+j)) \Rightarrow |\alpha(\varphi(6,m,j))| < \epsilon \]
\[ H((2,m,1)) \Rightarrow \text{Since } \epsilon \leq \frac{1}{6}, \text{ at least one product is made among the functions } \varphi(5,m,), \varphi(6,m,) \]

We can deduce: **Lemma:** any solution \((\text{APPROX})\) that we consider can be supposed to involve only 1 product for each pair of clusters.

**STEP 3:**

We claim that if solving this created \((\text{APPROX})\) problem leads to a cost \(c = Q + M\) then the corresponding \((3\text{-SAT})\) problem has a solution which can be reconstructed from the \(g_s\). Otherwise \(c\) will be \(> Q + M\) and there will be no solution to the corresponding \((3\text{-SAT})\). This will prove that \((\text{APPROX})\) is NP-hard.

At this step, we only prove the first direction:

**First direction:**

We notice that if there is a solution to the given \((3\text{-SAT})\) problem, then there is a solution to our constructed \((\text{APPROX})\) problem. For this, define for each \(q\) some \(G_p\) as \(\varphi(4,q,1) \times \varphi(4,q,2)\) or \(\varphi(4,q,1) \times \varphi(4,q,3)\) depending on whether \(X_q\) is true or false. Define also, for each \(m\), some \(G_p\) as \(\varphi(5,m,m') \times \varphi(6,m,j)\) where \(j\) is any (let’s say the first) value for which \(T_{m,j}\) is true, and \(m'\) is the number of times when we use \(\varphi(6,m,j)\) in such products. To approximate \(f\), we add all the \(G_p\) constructed ; that is we take \(\alpha_p = 1\ \forall p\).

This construction also gives a hint to the reader for guessing the way to recover the solution of \((3\text{-SAT})\) from the solution of \((\text{APPROX})\), by looking at which products were computed to construct the \(G_p\) s.

**STEP 4:**

**Other direction:**

Intuitively we can see that the product made on figure 3 should be \(\varphi(4,q,1) \times \varphi(4,q,2)\) or \(\varphi(4,q,1) \times \varphi(4,q,3)\) because if we try to approximate the value \(f((1,q,1)) = 1\), using \(\varphi(4,q,2) \varphi(4,q,3)\) would poorly approximate \(f\) at \((1,q,3)\). We will use this option to encode whether the boolean variable \(X_q\) is set to true \((\varphi(4,q,1)\varphi(4,q,2)\text{ is computed})\) or to false \((\varphi(4,q,1)\varphi(4,q,3)\text{ is computed})\).

Evaluations at \((1,q,)\) :

\[ H((1,q,3)) \Rightarrow |\alpha(\varphi(4,q,2) \times \varphi(4,q,3))| < \epsilon \]
\[ H((1,q,1)) \Rightarrow \text{Since } \epsilon \leq \frac{1}{5}, \text{ thanks to the Lemma: } \alpha(\varphi(4,q,1) \times \varphi(4,q,2)) \neq 0 \text{ xor } \alpha(\varphi(4,q,1) \times \varphi(4,q,3)) \neq 0 \]

Evaluation at \((2,m,)\) and \((3,m,m')\):

4
• $H((2, m, 2)) \Rightarrow |\alpha(\varphi(5, m, m') \times \varphi(5, m, m'))| < \epsilon$

• $H((2, m, 1)) \Rightarrow$ Since $\epsilon \leq \frac{1}{6}$, thanks to the Lemma: $\exists m', j : \alpha(\varphi(5, m, m') \times \varphi(6, m, j)) \neq 0$.

• $H((1, q, 4))$ (respectively $(1, q, 6)$) for $X_q$ (resp. $\overline{X}_q$) matching $T_{m,j}$ of the above constraints $\Rightarrow$ the product chosen in the previous analysis should match. That is, $\alpha(\varphi(4, q, 1) \times \varphi(4, q, 2)) \neq 0$, for this case where $T_{m,j} = X_q$ (resp. $\alpha(\varphi(4, q, 1) \times \varphi(4, q, 3)) \neq 0$, case where $T_{m,j} = \overline{X}_q$).

Therefore, fixing each $X_q$ to true or false depending on whether $\alpha(\varphi(4, q, 1) \times \varphi(4, q, 2)) \neq 0$ or $\alpha(\varphi(4, q, 1) \times \varphi(4, q, 3)) \neq 0$ gives a solution to the initial (3-SAT) problem.