On the minimum weight problem of permutation codes under Chebyshev distance

Min-Zheng Shieh  
Department of Computer Science  
National Chiao Tung University  
1001 University Road, Hsinchu, Taiwan  
Email: mzhsieh@csie.nctu.edu.tw

Shi-Chun Tsai  
Department of Computer Science  
National Chiao Tung University  
1001 University Road, Hsinchu, Taiwan  
Email: sctsai@cs.nctu.edu.tw

Abstract—Permutation codes of length $n$ and distance $d$ is a set of permutations on $n$ symbols, where the distance between any two elements in the set is at least $d$. Subgroup permutation codes are permutation codes with the property that the elements are closed under the operation of composition. In this paper, under the distance metric $\ell_\infty$-norm, we prove that finding the minimum weight codeword for subgroup permutation code is NP-complete. Moreover, we show that it is NP-hard to approximate the minimum weight within the factor $\frac{2}{3} - \epsilon$ for any $\epsilon > 0$.

I. INTRODUCTION

Permutation codes of length $n$ are subsets of all permutations over $\{1, \ldots, n\}$. We say a permutation code $C$ has minimum distance $d$ under some metric $\delta(\cdot, \cdot)$ if for any pair of distinct permutations $\pi$ and $\rho$ in $C$, $\delta(\pi, \rho) \geq d$. Recently, permutation codes have been found to be useful in several applications in various areas such as power line communication (see [12], [17], [18], and [16]), multi-level flash memories (e.g. [6], [7], and [14]), and cryptography (see [11]). For these applications, researchers mainly focus on creating permutation codes within certain distance $d$ under Hamming distance, Kendall’s tau distance, Chebyshev distance and other metrics which are meaningful for particular applications.

We use $S_n$ to represent all of the permutations over $\{1, \ldots, n\}$. $S_n$ is also called the symmetric group in Algebra. In this paper, we focus on permutation codes, which also form a subgroup of $S_n$. We call them subgroup codes. A subgroup code $C$ is often defined by a generator set $\{\pi_1, \ldots, \pi_k\}$ and all permutations in $C$ can be written in a sequence of compositions of elements in the generator set. This is similar to linear codes which are subspaces of $F^n$ for some finite field $F$ and positive integer $n$, and lattices which are subgroups of $\mathbb{R}^n$ under the vector addition for some positive integer $n$.

It is natural to ask how to determine the minimum distance of a code and to compute the closest codeword for a certain received string. Both problems have analogous versions for linear codes and lattices. For a linear code, it is to determine the minimum distance while given the generator matrix of the code. This problem under Hamming distance has been proved to be NP-complete by Vardy [15]. The analogous problem of the latter for linear codes under Hamming distance is also NP-hard by Arora et al [1]. The analogous problems for lattices are the shortest lattice vector problem (SVP) and closest vector problem (CVP). SVP under $\ell_p$-norm is NP-hard, even for approximating within $p^{1-\epsilon}$ for any $\epsilon > 0$ [3]. SVP under Chebyshev distance is also NP-hard, even for approximating within $n^{1/\log \log n}$ factor for any $\epsilon > 0$ [5]. For the subgroup permutation code version, both problems are proved to be NP-complete under many metrics, such as Hamming distance, $\ell_p$-norm, Kendall’s tau, etc [4], [3]. However, for Chebyshev distance ($\ell_\infty$-norm), the NP-completeness proof by Cameron and Wu [4] fell apart on some instances.

For right-invariant metrics, the minimum distance of subgroup permutation codes is equivalent to finding the minimum weight permutation $\pi$, where the weight of $\pi$ is defined as the distance between $\pi$ and the identity. In this paper, we focus on the complexity of the minimum weight problem for the subgroup permutation codes. We give a correct reduction to prove the NP-hardness of this problem. Moreover, we show that it is NP-hard to approximate within $\frac{2}{3} - \epsilon$ for any $\epsilon > 0$. Our result suggests that there does not exist an efficient method which can decide the minimum distance of an arbitrary subgroup permutation code. For example, in Tamo and Schwartz’s work [14], they constructed some subgroup permutation codes having a minimum distance larger than they proved, but they could not give the minimum distance explicitly with an efficient method. However, there are still some permutation codes coming with predetermined minimum distance, efficient encoding and decoding algorithms, such as in [10], [9] and [11]. The situation of subgroup permutation codes is just similar to linear codes. The rest of the paper is organized as follows. We define some notations in Section II. The reduction is given in Section III. Finally, Section IV concludes the paper.

II. PRELIMINARY

We use $[n]$ to indicate the set $\{1, \ldots, n\}$. A permutation $\pi$ over $[n]$ is a bijective function from $[n]$ to $[n]$. There are several representations for a permutation. In this paper, we use
a truth table to denote a permutation $\pi = [\pi(1), \ldots, \pi(n)]$, which can be written as the product of cycles. A cycle $(p_0, \ldots, p_{k-1})$ represents a permutation putting the $p_i$-th entry of the input to the $p_{i+1}$-th entry of the output for $i \in \mathbb{Z}_k$. Any permutation can be written in the form of product of disjoint cycles. For example, $\pi = [2, 3, 1, 4, 6, 5] = (1, 3, 2)(4)(5, 6)$. Usually, we ignore the cycles with only one element, therefore $[2, 3, 1, 4, 6, 5] = (1, 3, 2)(5, 6)$.

Let $S_n$ denote the set of all permutations over $[n]$. It is well known that $S_n$ is a group with the composition operation. We define the product of permutations $f$ and $g \in S_n$ as $fg = [f(g(1)), \ldots, f(g(n))]$. The identity permutation in $S_n$ is $e = [1, \ldots, n]$. We say that $\{\pi_1, \ldots, \pi_k\}$ is a generator set for a subgroup $H \subseteq S_n$, if every permutation $\pi \in H$ can be written as a product of a sequence of compositions from elements in the generator set. For two permutations $\pi$ and $\rho$ over $[n]$, their Chebyshev distance is defined as $\ell_\infty(\pi, \rho) = \max_{i \in [n]} |\pi(i) - \rho(i)|$. Note $\ell_\infty$ is a right-invariant metric, i.e., for permutations $\pi, \rho$, and $\tau$, we have $\ell_\infty(\pi \tau, \rho \tau) = \ell_\infty(\pi, \rho)$.

We say that a permutation $\pi$ has weight $w$ under right-invariant metric $\delta$ if $\delta(e, \pi) = w$. Now we define the minimum weight problem of subgroup permutation code under Chebyshev metric, and we call it MINWSPA for short.

**Definition 1. (MINWSPA)** Given a generator set $\{g_1, \ldots, g_k\}$ for a subgroup $H$ of $S_n$ and an integer $B$, determine if there exists a permutation $\pi \in H$ that has a non-zero weight $w \leq B$.

Klein four-group is the building block of our proofs. It is defined as $K_4 = \{e, \kappa_1, \kappa_2, \kappa_3\}$, where $\kappa_1 = (1, 2)(3, 4)$, $\kappa_2 = (1, 3)(2, 4)$, and $\kappa_3 = (1, 4)(2, 3)$. Its operation is shown in Table I. It is clear that $K_4$ is commutative and $\ell_\infty(e, \kappa_i) = i$ for $i \in \{1, 2, 3\}$. We also use shift and stretch operations for constructing permutations. They may involve some elements of large indices. We assume that these operations are only applied on permutations over a sufficiently large symbol set. Shifting a cycle $(p_1, \ldots, p_k)$ is to add the same number to each entry of it. For example, if we shift $(1, 2, 3)$ with 5, then we get $(6, 7, 8)$. We denote the shift operation as $s_r(\pi)$, which shifts all cycles in $\pi$ with the number $r$. For example $s_4(\kappa_1) = (1, 4)(2, 4)(3, 4, 5)(6, 7, 8)$. This operation does not change the weight since the distance is preserved.

Stretching a cycle $(p_1, \ldots, p_k)$ is to multiply each entry by the same number. For example, if we stretch $(1, 2, 3)$ by 2 then we have $(2, 4, 6)$. We denote the stretch operation as $a_t(\pi)$ which stretches all cycles in $\pi$ by the number $t$ and then shifts the cycles such that the smallest symbol is down to 1. The distance is amplified $t$ times, and so is the weight of the cycles. For example $a_2(\kappa_1) = s_{-1}(1(1, 2, 2, 4)(3, 2, 4, 2)) = (1, 3)(5, 7)$, and similarly $a_2(\kappa_2) = (1, 5)(3, 7), a_2(\kappa_3) = (1, 7)(3, 5)$. This operation amplifies the weight 2 times. Observe that if $\{i, j, k\} = \{1, 2, 3\}$, then $s_r(a_i(\kappa_1))s_r(a_j(\kappa_2)) = s_r(a_i(\kappa_2))$ and $s_r(a_i(\kappa_3))s_r(a_j(\kappa_3)) = s_r(a_i(e))$. I.e., the shift and stretch operations preserve the property of Klein four-group.

### III. Reduction

In this section we give a reduction from Not-All-Equal-SAT (NAESAT) to MINWSPA. Cameron and Wu[4] gave a proof by a reduction from NAESAT to MINWSPA, but their construction fell apart on $\ell_\infty$-norm for some instance, which is shown in Appendix A. We give the formal definition of Not-All-Equal-SAT problem as follows.

**Definition 2. (NAESAT)** Given a boolean formula $\phi$ in conjunctive normal form, which consists of $m$ exact-3-literal clauses $c_1, \ldots, c_m$ of over $n$ variables $x_1, \ldots, x_n$, decide whether there exists an assignment $\sigma$ such that for every clause $c$, not all literals in $c$ are assigned to the same truth value.

To construct the corresponding generator set from an NAESAT instance $\phi$, we define three kinds of permutation gadgets for the clauses, variables and the truth assignment over $[48m + 18n]$. Our goal is mapping truth assignments for $\phi$ to permutation codewords in the corresponding subgroup permutation code. Moreover, the codewords converted from satisfying assignments have less weight than the other codewords, except the identity. Hence, we can determine whether $\phi$ is satisfiable from the minimum weight of the corresponding subgroup permutation code.

The clause gadgets permute $1, \ldots, 48m$, which are derived from the work by Cameron and Wu[4]. The main idea of the clause gadget is to assure that all literals are not assigned to the same value. For convenience, we also express the following permutations with the shift and stretch operations. Let $h_1 = (1, 3)(5, 7)(2, 4)(6, 8)(9, 13)(11, 15)(10, 14)(12, 16)\ldots (17, 23)(19, 21)(18, 24)(20, 22)$

$$= a_2(\kappa_1}s_4(a_2(\kappa_1))s_8(a_2(\kappa_3))s_9(a_2(\kappa_2))s_{16}(a_2(\kappa_2)s_{17}(a_2(\kappa_2))) \]

$$h_2 = (1, 5)(3, 7)(2, 6)(4, 8)(9, 15)(11, 13)(10, 16)(12, 14)\ldots (17, 19)(21, 23)(18, 20)(22, 24)$$

$$= a_2(\kappa_2)s_4(a_2(\kappa_2))s_8(a_2(\kappa_3))s_9(a_2(\kappa_3))s_{16}(a_2(\kappa_1)s_{17}(a_2(\kappa_1))) \]

$$h_3 = (1, 7)(3, 5)(2, 8)(4, 6)(9, 11)(13, 15)(10, 12)(14, 16)\ldots (17, 21)(19, 23)(18, 22)(20, 24)$$

$$= a_2(\kappa_3)s_4(a_2(\kappa_3))s_8(a_2(\kappa_1))s_9(a_2(\kappa_1))s_{16}(a_2(\kappa_2)s_{17}(a_2(\kappa_2))) \]

$$g = (1, 8)(2, 7)(3, 6)(4, 5)(9, 16)(10, 15)(11, 14)(12, 13)\ldots (17, 24)(18, 23)(19, 22)(20, 21)$$
Let $H$ be the subgroup generated by $\{g_i, g'_i : i \in [n]\}$, i.e., $H = \langle g_i, g'_i : i \in [n] \rangle$. We can obtain a permutation

$$\pi = \left( \prod_{\sigma(\xi_i) = T} g_i \right) \left( \prod_{\sigma(\xi_i) = F} g'_i \right) \in H$$

from an assignment $\sigma$ for $\phi$. By the following two lemmas, we show that there must exist a non-identity permutation of minimum weight which is constructed from an assignment.

Lemma 1. The permutations mapped from satisfying assignments have weight 6 and the permutations mapped from unsatisfying assignments have weight 7.

Proof:
Let $\pi$ be the permutation obtained from an assignment $\sigma$ of $\phi$. Note that the elements permuted by the clause gadgets, variable gadgets, and assignment gadgets are disjoint. Therefore we can discuss the weight of them separately in three categories. First, we look at the elements permuted by variable gadgets $v_i,T$ and $v_i,F$ for $i \in [n]$. Only $g_i$ and $g'_i$ can alter these elements, and $\pi$ has exactly one of them. Thus, the difference between $\pi$ and $e$ on these elements is at most 6. Next we turn to assignment gadget $u_i$ for some $i \in [n]$. Without loss of generality, we assume $u_i = (n_1, n_1 + 7)(n_2, n_2 + 7)$, $u_{i-1}$ contains $(n_1, n_1 + 7)$ and $u_{i+1}$ contains $(n_2, n_2 + 7)$. Since $\pi$ has exactly one of $g_i$ and $g'_i$ for every $i \in [n]$, both $(n_1, n_1 + 7)$ and $(n_2, n_2 + 7)$ appear twice in the construction of $\pi$. Moreover, they are the only cycles covers $n_1, n_1 + 7, n_2$, and $n_2 + 7$. Thus, $\pi$ does not affect these elements and there is no difference between $\pi$ and $e$ on them.

At last, we observe clause gadgets $h_{j,k,T}$ and $h_{j,k,F}$. We claim that for $j \in [m]$ and $k \in [3]$, exactly one of $h_{j,k,T}$ and $h_{j,k,F}$ appears in $\pi$. Assume $x_j$ is the $k$-th literal of the $j$-th clause. If $\sigma(x_j) = T$, then $h_{j,k,T}$ is picked by the definition of $g_i$, otherwise $\pi$ picks $h_{j,k,F}$. It is similar for the case that $\bar{x}_j$ is the $k$-th literal in the $j$-th clause. Thus, for every $j \in [m]$, $\pi$ picks exactly three out of $h_{j,1,T}, h_{j,2,T}, h_{j,3,T}$, $h_{j,1,F}, h_{j,2,F}$, and $h_{j,3,F}$. Let $A_j = \{h_{j,k,T} : k \in [3]\}$ and $B_j = \{h_{j,k,F} : k \in [3]\}$. In the following, we discuss how these gadgets affect the distance between $\pi$ and $e$.

1) If the gadgets in $A_j$ are not picked at all, then the elements permuted by $A_j$ remain the same as $e$. But this implies all gadget in $B_j$ are picked, then the elements permuted by $B_j$ are permuted with a shift of $g$. The distance is $\max\{0, 7\} = 7$.
2) If one of $A_j$ and two of $B_j$ are picked, then the elements permuted by $A_j$ and $B_j$ are in the form of $g h_k$ and $h_{k'}$ for some $k, k' \in [3]$, respectively. The distance is $\max\{5, 6\} = 6$.
3) If two of $A_j$ and one of $B_j$ are picked, then, similar to 2), the distance is 6.
4) If all of $A_j$ are picked, then, similar to 1), the distance is 7.

Note that the first or last cases above happen if and only if $\sigma$ is not a satisfying assignment. Since distances of the clause
gadgets dominate the distance over the other gadgets, we conclude that \( \pi \) has weight 6 if \( \sigma \) is satisfying; 7 otherwise.

Lemma 2. The other non-identity permutations in \( H \) have weight at least 7.

Proof: Since all gadgets are commutative, we can express any permutation \( \pi \in H \) as a product of generators, i.e., \( \pi = g_{i_1}^z(g_{i_1}')^z_1 \cdots g_{i_n}^z(g_{i_n}')^z_n \). Since every gadget is the inverse of itself, we assume \( z_1, z_1', \ldots, z_n, z_n' \in \{0, 1\} \) without loss of generality. A permutation converted from an assignment must choose either \( g_i \) or \( g_i' \), for every \( i \in [n] \), i.e., \( z_i + z_i' = 1 \) for \( i \in [n] \). So we discuss the following two cases.

- If there exists some \( i \) such that \( z_i + z_i' = 2 \), then \( \pi \) picks both \( g_i \) and \( g_i' \). In this case, the elements corresponding to \( v_{i,F} \) and \( v_i'F \) are permuted into the form of \( v_{i,F}v_{i,F} \), which has weight 9.
- For every \( i \in [n], z_i + z_i' \neq 2 \). Because \( \pi \not\in \epsilon \) and \( \pi \) is not converted from an assignment, there are \( i_0 \) and \( i_1 \) such that \( z_{i_0} + z_{i_0}' = 0 \) and \( z_{i_1} + z_{i_1}' = 1 \). Recall that for \( i < n \), \( u_i = z_{b_2 + 8(i - 1)}(1,8)z_{b_2 + 8(1,8)} \) and \( u_n = z_{b_2 + 8n}(1,8)z_{b_2}(1,8) \). Without loss of generality, we can assume that \( i_0 = i_1 = 1 \). As a consequence, \( u_{i_0} \) and \( u_{i_1} \) are the only two gadgets permuting \( b_2 + 8i_1 - 7 \) and \( b_2 + 8i_1 + 7 \). Since \( z_{i_0} + z_{i_0}' + z_{i_1} + z_{i_1}' = 0 + 1 = 1 \), \( \pi \) picks exactly one of \( g_{i_0}, g_{i_0}' \), \( g_{i_1} \), and \( g_{i_1}' \). \( b_2 + 8i_1 - 7 \) and \( b_2 + 8i_1 + 7 \) must be swapped by \( \pi \), hence \( \pi \) has distance at least 7 in this case.

The non-identity permutations, which are not in the two cases, have exactly one of \( g_i \) and \( g_i' \) for every \( i \in [n] \), and these can be obtained from assignments. We conclude the lemma is true.

With the two lemmas above, we prove the following theorem.

Theorem 1. Let \( H \) be the group generated by \( \{g_1, g_1', \ldots, g_n, g_n'\} \) which is mapped from a NAESAT instance \( \phi \). If \( \phi \) is satisfiable then \( H \) has minimum weight 6, otherwise \( H \) has minimum weight 7.

Proof: For satisfiable \( \phi \), there exists a satisfying assignment \( \sigma \). We can convert \( \sigma \) into \( \pi \), and \( \pi \) has weight 6 by lemma [1]. For unsatisfiable \( \phi \), all assignments are not satisfying. So every permutation converted from an assignment has weight 7, and the other non-identity permutations have weight at least 7. Thus, we conclude \( H \) has minimum weight 7.

From the above theorem we have an immediate inapproximable result. We say that an algorithm \( A \) is an \( r \)-approximate algorithm for a minimization problem if \( A \) always outputs a feasible solution whose cost is no more than \( r \) times of the minimum cost on any input. Note that \( A \) cannot output an answer whose cost is less than the minimum cost, since it is not a feasible solution. Since NAESAT is an NP-complete problem, we have the following corollary as an immediate result of theorem [1].

Corollary 1. MINWSPA is NP-complete. Moreover, it is NP-hard to approximate within \( \frac{7}{6} - \epsilon \) for any \( \epsilon > 0 \).

Proof: MINWSPA is in NP, since we can finish computing the weight of any permutation \( \pi \) and verifying if \( \pi \) is in the subgroup by Schreier-Sims algorithm [3] in polynomial time. By theorem [1] \( \phi \) is satisfiable if and only if the corresponding subgroup \( H \) has minimum weight at most 6. Hence, we can conclude MINWSPA is NP-complete. Now, assume we have a polynomial time \( \left( \frac{7}{6} - \epsilon \right) \)-approximate algorithm \( A \). We can construct a polynomial time algorithm to solve NAESAT.

1) Construct the subgroup \( H \) from \( \phi \) and run \( A(H) \).
2) If \( A(H) \) outputs a number no more than \( 7 - 6\epsilon \), then accept \( \phi \), otherwise reject.

Any satisfiable \( \phi \) will be accepted, and all unsatisfiable \( \phi \)'s will be rejected, since an approximate algorithm cannot give an answer less than the minimum solution which is 7.

IV. CONCLUSION

We show that MINWSPA is NP-complete. It implies that the minimum weight problem of permutation codes under the well known metrics are all NP-complete. For the case of \( \ell_\infty \)-metric, we also prove that there is no \( \left( \frac{7}{6} - \epsilon \right) \)-approximate algorithm for any \( \epsilon > 0 \) unless \( \text{P}=\text{NP} \). We believe that the minimum weight problems under other metrics also have inapproximable results, however, they still remain open. Our inapproximable result still has room for improvement. It is interesting to find better approximation algorithm with some constant \( \epsilon > \frac{1}{6} \).

REFERENCES

[1] S. Arora, L. Babai, J. Stern, Z. Sweedyk, “The Hardness of Approximate Optima in Lattices, Codes, and Systems of Linear Equations,” Journal of Computer and System Science, vol. 54, 1997, pp. 317-331.
[2] E. R. Berlekamp, R. J. McEliece, H. C.A. van Tilborg, “On the inherent intractability of certain coding problems,” IEEE Trans. Inform. Theory IT-24, 1978, pp. 384-386.
[3] C. Buchheim, P. J. Cameron, T. Wu, “On the subgroup distance problem,” Discrete Mathematics, vol. 309, pp. 962-968, 2009.
[4] P. J. Cameron, T. Wu, “The complexity of the weight problem for permutation and matrix groups,” Discrete Mathematics, vol. 310, pp. 408-416, 2010.
[5] I. Dinur, “Approximating SVP_∞ to within almost polynomial factors is NP-hard,” Combinatorica, vol. 23, 2003, pp. 205-243.
[6] A. Jiang, R. Mateescu, M. Schwartz, J. Bruck, “Rank Modulation for Flash Memories,” in Proc. IEEE Internat. Symp. on Inform. Th., 2008, pp. 1731-1735.
[7] A. Jiang, M. Schwartz, J. Bruck, “Error-Correcting Codes for Rank Modulation,” in Proc. IEEE Internat. Symp. on Inform. Th., 2008, pp. 1736-1740.
[8] S. Khot, “Hardness of Approximating the Shortest Vector Problem in Lattices,” J. ACM, Vol. 52, No. 5, 2005, pp. 789-808.
[9] T. Klove, T.-T. Lin, S.-C. Tsai, W.-G. Tzeng, “Permutation arrays under the Chebyshev distance,” in Proc. IEEE Trans. on Inform. Th., accepted and to appear, 2010.
[10] T.-T. Lin, S.-C. Tsai, W.-G. Tzeng, “Efficient Encoding and Decoding with Permutation Arrays,” in Proc. IEEE Internat. Symp. on Inform. Th., 2008, pp. 211-214.
APPENDIX

A. Cameron-Wu’s reduction

The reduction in Cameron and Wu’s work [4] uses only two kinds of gadgets. The variable gadget \( v_i \) for the \( i \)-th variable is \( (2i-1, 2i) \). The clause gadget \( h_{j,k} \) for the \( k \)-th literal in the \( j \)-th clause is defined as \( s_{2n+24(j-1)}(h_k) \) where \( h_1, h_2, h_3 \) and \( g \) are the same as in this paper. The generators are defined as

\[
g_i = v_i \left( \prod_{(i,j,k) \in P} s_{2n+24(j-1)}(h_k) \right),
\]

\[
g'_i = v_i \left( \prod_{(i,j,k) \in Q} s_{2n+24(j-1)}(h_k) \right),
\]

where \( P, Q \) are the same sets as in our reduction. They also construct a generator \( g_c = \prod_{j \in [m]} s_{2n+24(j-1)}(g) \) acting as \( g \) on every clause gadget. Their construction does not work in the following instance. Let \( \phi = (x_1 \lor x_2 \lor x_2) \land (\bar{x}_1 \lor x_2 \lor x_2) \).

By their construction, subgroup \( G \) is generated by

\[
g_1 = v_1 h_{1,1} = (1, 2) s_4(h_1),
\]

\[
g'_1 = v_1 h_{2,1} = (1, 2) s_{28}(h_1),
\]

\[
g_2 = v_2 h_{1,2} h_{1,3} h_{2,2} h_{2,3} = (3, 4) s_4 h_2 s_4 h_3 s_{28}(h_2) s_{28}(h_3) = (3, 4) s_4 h_1 s_{28}(h_1),
\]

\[
g'_2 = v_2 = (3, 4),
\]

\[
g_c = s_4(g) s_{28}(g).
\]

Note that \( \phi \) is an unsatisfiable formula for NAESAT. According to their proof of Theorem 18[4], elements of \( G \) should not have weight 5, since \( \phi \) is unsatisfiable. But \( g_2 g'_2 g_1 g'_1 = s_4(gh_1) s_{28}(gh_1) \) has weight 5. Therefore, we need to design the gadgets more carefully to prove that MINSWPA is NP-complete.