On neural circuits of working memory sequence permutation: optimizing circuit architectures via Cayley graphs

Kevin Bien KEVIN.BIEN@TCU.EDU
Texas Christian University, Fort Worth, Texas

Junfeng Zuo ZUOJUNFENG@PKU.EDU.CN
Peking University, Beijing, China

Wen-Hao Zhang WENHAO.ZHANG@UTSOUTHWESTERN.EDU
UT Southwestern Medical Center, Dallas, Texas

Editors: List of editors' names

Abstract

The brain's ability to store and manipulate working memory (WM) sequences is pivotal for higher cognitive reasoning. Although the neural circuit mechanisms for storing WM sequences have been extensively studied, those for manipulating WM sequences remain largely unknown. Inspired by a recent WM sequence manipulation experiment in monkeys, we design a functional, biologically plausible neural circuit model that realizes WM sequence permutations using guidance from permutation groups and their Cayley graph representations. The circuit consists of two interconnected modules: a memory module composed of continuous attractor-based memory motifs that store and interchange items in WM sequences, and a control module that sends gain modulations to guide permutation operations within the memory module. The control module features a hierarchical tree structure that decomposes complex permutations into a sequence of basic two-item swaps, simplifying circuit implementations. We demonstrate that permutation circuit architectures have one-to-one correspondence with Cayley graphs representing permutation group structure, where the group generating set directly determines the connectivity between memory motifs. Since each permutation group may have multiple generating sets, there are multiple circuit architectures implementing the same permutation. We therefore utilize Cayley graph analysis to determine trade-offs between computational efficiency, circuit complexity, and circuit robustness. Our study establishes connections between abstract group theory, Cayley graphs, and biologically plausible circuit architectures, providing insights into principled circuit design via algebraic frameworks.

1. Introduction

Working memory (WM) is one of the central components of cognitive reasoning (Fuster and Alexander, 1971; Miller and Cohen, 2001), which includes **maintaining** information temporarily and **manipulating** this information based on some rules. Studying how neural circuits in the brain implement WM maintenance and manipulation can advance our understanding of the neural basis of cognitive reasoning and may inspire next-generation AI systems. While WM research has traditionally focused on maintenance, recent experimental and theoretical studies have begun to investigate WM manipulation. An elementary operation of WM manipulation is re-ordering WM sequences (Stephens, 1963; Tarr and Pinker, 1989; Miller et al., 2018; Tian et al., 2024), e.g., swapping two WM items. A recent

neuroscience experiment (Tian et al., 2024) provided unprecedented electrophysiological observations on the neural dynamics underlying this operation. It suggests that the frontal cortex explicitly swaps the WM items stored in two memory subspaces using a parallel swapping algorithm with two steps: loading the content from two memory subspaces to corresponding temporary subspaces, then cross loading the content from temporary subspaces back to memory subspaces (Fig.1A). Building on this finding, a circuit model (Zuo et al., 2025) was developed to maintain and manipulate WM sequences. The model consists of two interconnected modules: a memory module composed of continuous attractor network (CAN)-based (temporary) memory subspaces storing WM item features and a control module issuing gain-modulation commands to orchestrate specific operations in the memory module. The circuit successfully swaps the two items in the WM sequence by utilizing autonomous circuit dynamics and generates neuronal responses consistent with experimental observations. Furthermore, the model was extended to permute 3-item WM sequences by executing two swaps in sequence.

The present study focuses on the scalability of the permutation operation. Given a set of basic operations, we ask whether arbitrarily complex operations can be efficiently executed through their proper composition. Specifically, by strategically selecting which items to swap, an n-item permutation (for large n) can be constructed with a minimal number of swaps. Because such compositions are naturally captured by the symmetric (permutation) group, we assess the scalability of different circuit architectures by examining their corresponding Cayley graph representations. This approach allows us to identify which architectures most efficiently and robustly manipulate longer WM sequences. Our analysis suggests that optimal performance is achieved when a small subset of memory subspaces function as relay centers—that is, when the memory module exhibits a community structure—so that items stored in WM can be flexibly routed to and from these hubs. To demonstrate the viability of our approach, we develop and simulate a circuit capable of manipulating sequences of up to 6 items.

2. A neural circuit model for WM swapping and permutation

The swap can be implemented by a neural circuit model governed by autonomous dynamics. This model is built from well-established canonical neural circuit motifs and operations, which are organically integrated to realize swaps between WM items. The detailed math of the circuit dynamics is in App. A.

2.1. Overview of circuit architecture for swapping

The circuit has two recurrently connected modules (Fig. 1B-C): A **memory** module for storing information, and a **control** module for manipulating WM contents. The control module sends gain modulation to excite the memory module, while the memory module sends inhibitory feedback to the control module to register the execution of an operation.

Mimicking typical WM experiments (e.g., Tian et al. (2024)), each WM item is a continuous feature like orientation and needs to be stored in the memory module circuit. In the present study, the memory circuit storing one WM item is modeled as a continuous attractor network (CAN), a canonical circuit model in neuroscience that represents continuous features (Compte et al., 2000; Wimmer et al., 2014; Ben-Yishai et al., 1995; Knierim and

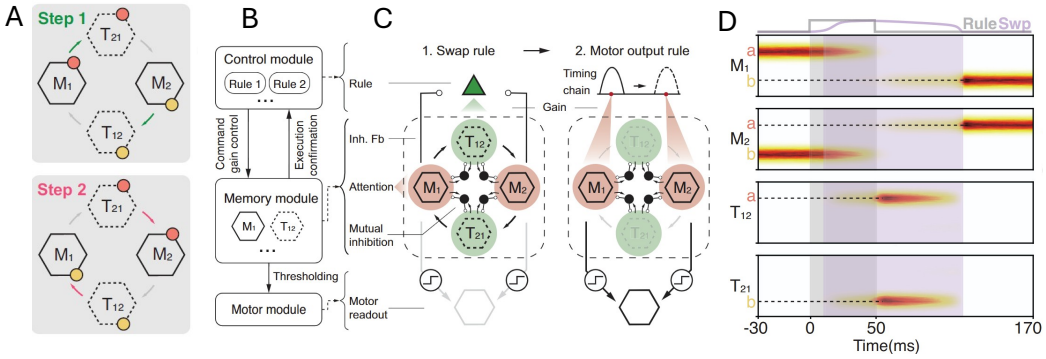

Figure 1: (A) Diagram of the parallel swap algorithm shown to be used in primate WM. (B-C) The modular structure of the recurrent circuit for swapping and outputting two WM items. (D) Population activities of (temporary) memory motifs during swapping. Shaded areas indicate the time windows when the swapping cue is presented (gray) and swapping control neuron fires (purple). Adapted from Zuo et al. (2025).

Zhang, 2012; Wu et al., 2016; Khona and Fiete, 2022). A CAN consists of feature-selective excitatory (E) neurons and a pool of inhibitory (I) neurons. The E neurons utilize their structured recurrent kernel to give rise to structured population responses to store WM items (Eq. 6), while I neurons stabilize the network dynamics. Importantly, the gain of E neurons can be modulated by the control circuit (Eq. 8) that switches a CAN between *Up* (high response) and *Down* (low response) states via the cusp bifurcation (Guckenheimer and Kuznetsov, 2007). We will show the control circuit can utilize the gain modulation to command the memory module to execute corresponding operations.

Different CANs are interconnected through recurrent connection kernels. To execute a swap, it requires two memory motifs (M_1 and M_2) to store the items and two temporary motifs (T_{12} and T_{21}) to hold the items in intermediate steps. These motifs form a circle structure in connection topology ($M_1 \to T_{12} \to M_2 \to T_{21} \to M_1$, Fig. 1C). I neurons associated with each CAN are driven by E neurons in the same and neighboring CANs (Eq. 9). The shared inhibition between CANs introduces competitions between them, which is crucial for autonomous swapping.

The control circuit also has recurrent dynamics consisting of E and I neurons. When receiving external swapping signals, it provides gain modulations to E neurons in the temporary motif, and meanwhile receives feedback from memory motifs. The mutual interactions between the control and the memory modules are crucial for utilizing the recurrent circuit's autonomous dynamics to perform swapping while rule cues are transiently presented.

The swapping process is coordinated by gain modulation from the control circuit (Fig. 3B). The key principle is that gain modulation can flexibly switch the flow of information on or off without altering synaptic weights. Memory items were initially stored in memory motifs M_1 and M_2. Once the swapping process is initiated, the control circuit will be activated to increase the gain of temporary motifs (T_{12} and T_{21}) and switch them on, corresponding to loading the content from memory motifs. Meanwhile, content in memory motifs will be erased by mutual inhibition. The swapping control neuron provides sustained gain to temporary motifs, enabling temporary motifs to reactivate subsequent memory

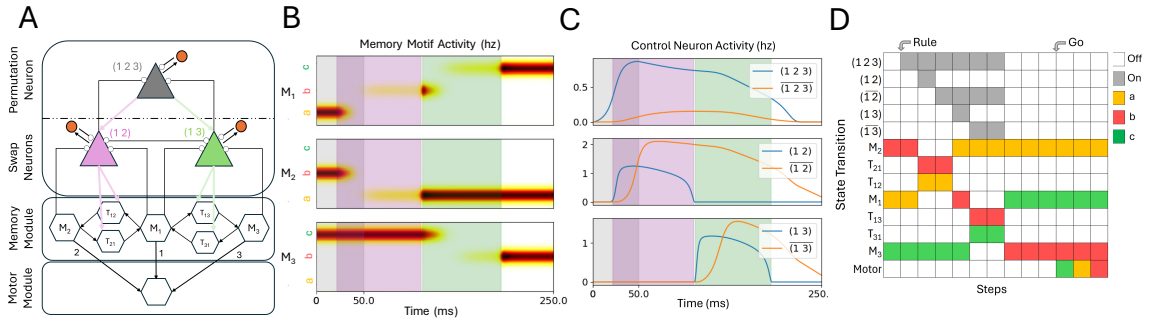

Figure 2: (A) Diagram of 3-item permutation circuit. The gain input from the 3-permutation to swapping control neurons decreases with the order of swaps in the sequence. Triangles are permutation/swap control neurons; orange circles are conjugate neurons. (B) Neural activity of memory motifs. (C) Neural activity of control neurons. (E) State transition matrix illustrating component activity across time. Adapted from Zuo et al. (2025).

motifs and relay WM contents to them, completing the WM item swapping. In the end, control neuron will be shut down by inhibitory feedback from memory motifs after they rebuild their activities, confirming that the swapping operation is executed.

2.2. From swapping to sequence permutation

Permutation group theory can be used to describe the underlying structure of permutations. The set of 3-item permutations is represented by the group S_3 containing 6 permutations: {(),(1 2),(2 3),(1 3),(1 2 3),(1 3 2)} (using the cycle notation). () is the identity operation that does not change the sequence, (1 2) denotes swapping the first and second items, i.e., changing the sequence *ab* into *ba*. (1 2 3) denotes moving position 1's original item to position 2, position 2's original item to position 3, and position 3's original item to position 1, i.e., changing the original sequence *abc* to *cab*.

Any permutation can be generated through a sequence of elementary swaps. For example, (1 2 3) can be decomposed as:

$$(1\ 2\ 3) = (1\ 3)(1\ 2) : abc \xrightarrow{(1\ 2)} bac \xrightarrow{(1\ 3)} cab, \tag{1}$$

where the order of the two-swap product defines sequential execution. Note that this decomposition is not unique, e.g., (1 2 3) can also be decomposed as (2 3)(1 3) or (1 2)(2 3).

2.3. A neural circuit for 3-item sequence manipulation

Inspired by the structure of the permutation group S_3 (Eq. 1), a tree-structured control module can be developed to extend the swapping circuit to perform 3-item permutations: each 3-item permutation (represented by a parent node) can be orchestrated by composing multiple 2-item swaps (represented by child nodes).

Child nodes (Fig. 2A, triangles) execute individual swaps by modulating the gain sent to the memory module, while parent nodes modulate the gain sent to child nodes to execute a sequence of swaps. To ensure that swaps are executed in order, the top-down modulations

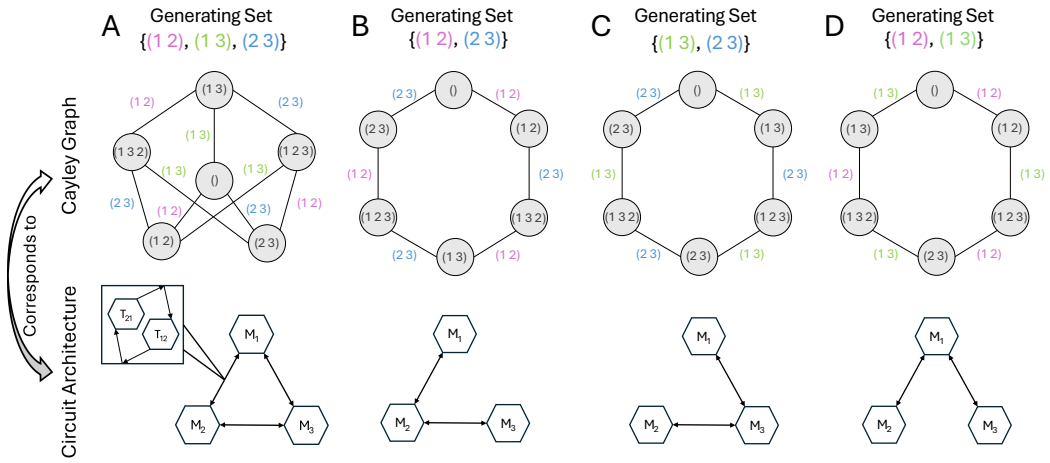

Figure 3: The Cayley graph of S_3 based on the generating set (shown at the top) corresponds to the connectivity between memory modules. Top row: Cayley graphs where each node represents a permutation and each bidirectional edge represents the swap converting the permutations at its two ends. Bottom row: the corresponding circuit architecture implementing the denoted generating set. Each swap in the generating set requires that corresponding memory slots are connected via two temporary slots, e.g., (1 2) requires M_1 and M_2 to be connected via T_{12} and T_{21}.

on the tree has a spatial gradient over the child nodes. Nodes on the same level will mutually inhibit each other, preventing them from interference. Similar to the swapping circuit, child nodes feed back an inhibitory signal to their parent nodes to confirm the execution of swaps. Furthermore, each neuron in the control tree is paired with a conjugate neuron (Fig. 2A, orange circles) that provides inhibitory feedback. The conjugate neuron is activated once the corresponding swap is executed, and remains so until the completion of the whole permutation (controlled by the root node). The conjugate neurons counteract redundant gain on the already executed swaps, instantiating an 'inhibition of return'.

While there is not yet empirical evidence that neural systems manipulate WM sequences by composing basic swaps, our model makes direct predictions about the activity of memory subspaces during a sequence manipulation task (Fig. 2B), and could be validated through existing experimental paradigms (Tian et al., 2024).

3. Comparing circuit architecture efficiency

Directly swapping the contents of two memory motifs requires the use of two temporary motifs, which are gain-modulated by the control circuit. This process consumes both time and storage resources. However, as discussed in the previous section, the decomposition of permutations is not unique. By carefully selecting which swaps to execute, the system can significantly reduce the average cost associated with implementing any possible permutation.

3.1. Analyzing the circuit architecture via the group generating set

A set of swaps from which we can compose all permutations in a permutation group is called a **generating set**. Given a permutation group S_n, there can be multiple generating sets. For example, S_3 has four generating sets of swaps: $\{(1\ 2), (1\ 3), (2\ 3)\}$, $\{(1\ 2), (2\ 3)\}$, $\{(1\ 3), (2\ 3)\}$, and $\{(1\ 2), (1\ 3)\}$. Permutations in S_3 are are produced by elements in the generating sets through function composition. One way to visualize the group structure given the choice of a generating set $G \subset S_n$ is the **Cayley graph** $C(S_n, G)$ (Fig. 3). Each node of $C(S_n, G)$ represents a permutation in S_n, and two nodes are connected if one can be transformed into another by applying a single swap from set G, represented by an edge. A decomposition of a permutation is given by the path between it and the identity node. By tracing the *shortest path* from each node (a permutation) to the identity node (), we can determine the optimal decomposition of each 3-item permutation under the choice of generating set. For example, we can see that the permutation node (1 2 3) is connected to () through the path $((1\ 2), (1\ 3))$ (Fig. 3A), visualizing Eq. (1).

Different generating sets allow different decomposition strategies. A larger generating set can decompose permutations in fewer steps with higher computational efficiency. For example, every node in $C(S_3,\ \{(1\ 2), (1\ 3), (2\ 3)\})$ (Fig. 3A) is no more than 2 edges away from (), but all other graphs corresponding to smaller generating sets have a permutation with longer paths of 3 edges (Fig. 3B-D). Therefore, by choosing a large generating set, we can minimize the number of swaps executed to implement each permutation, in turn improve the computation efficiency. However, at the circuit level, implementing one swap in a generating set requires two temporary motifs connecting a pair of memory motifs, so a larger generating set requires more temporary motifs and increased circuit complexity. Thus there is a trade-off between circuit complexity and computational efficiency.

3.2. Balancing circuit complexity and computational efficiency

For the circuit to decompose every permutation in S_n into the minimal number of swaps, it would need to implement the full set of swaps, which would require $2 \times \binom{n}{2}$ temporary motifs to implement. To determine whether another memory module architecture could decompose permutations nearly as efficiently without requiring as many temporary motifs, we examined how the average number of swaps needed to decompose a permutation in S_n varied across different generating sets. We can compute this using Cayley graphs as:

$$\bar{d}(G, n) = \frac{1}{n!} \sum_{v} d(v, ()), \quad v \in C(S_n, G), \tag{2}$$

where $d(v, ())$ is the minimal number of edges between the node v and the identity () on $C(S_n, G)$. We first consider three example generating sets with corresponding circuit architectures shown in Fig. 4A: the full set of swaps (denoted as F), the set of swaps between adjacent motifs (denoted as A), and a centralized set composed of $(1\ i)$, $i \in \{2, ..., n\}$ (denoted as C). Fig. 4B plots $\bar{d}(G, n)$ across each of these generating sets. The average decomposition length was computed for permutation groups up to S_8, whose sequence length is longer than the typical working memory capacity, ranging from 4 to 7 (Luck and Vogel, 1997; Cowan, 2001).

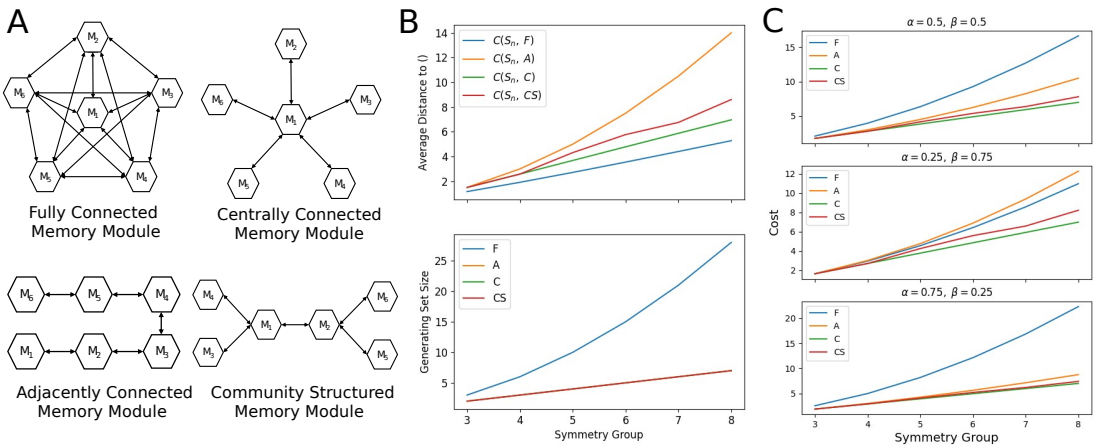

Figure 4: (A) Diagrams of memory modules implementing different generating sets of S_6. (B, top) Average distance to () on Cayley graphs across different generating sets ($\bar{d}(G,n)$) for $3 \leq n \leq 8$. (bottom) Sizes of different generating sets ($|G|$) for $3 \leq n \leq 8$. (C) Cost(G,n) across different generating sets and weights for $3 \leq n \leq 8$

To determine which generating set achieves the best balance between time and memory efficiency, we define a cost function:

$$\text{Cost}(G,n) = \alpha|G| \ + \ \beta\bar{d}(G,n) \tag{3}$$

where α and β are trade-off coefficients, and $|G|$ denotes the size of the generating set. We found that across a wide range of coefficients ($\alpha, \beta \in [0.25, 0.75]$), the centralized architecture has the minimal cost (Fig. 4C), suggesting the circuit obtains an optimal balance between circuit complexity and computational efficiency when it relays activity between a central memory motif (a "relay motif", M_1 in this case) and every other memory motif.

3.3. Improving the circuit robustness via a community structured memory module

Although the centralized architecture achieves a balance between time and memory efficiency, it has a severe vulnerability: if the relay motif is dysfunctional, the circuit will not be able to produce any permutation. To mitigate this vulnerability, we can rearrange the memory module so that motif connections are distributed across multiple relay motifs. Each relay motif is a locally centralized node forming a local community, while relay motifs across communities are connected, creating a **community structured** network. Such an architecture can emerge organically through a simple preferential attachment mechanism (see Alg. 1).

By incorporating multiple relay motifs, we can ensure that losing one relay motif only impairs permutations for the corresponding community while leaving other communities intact. This implies that the community structured architecture is less vulnerable to failure from a single component and is more robust.

To quantify the robustness, we measure the minimum number of permutations the memory module can still execute after inactivating one memory motif. To formalize this,

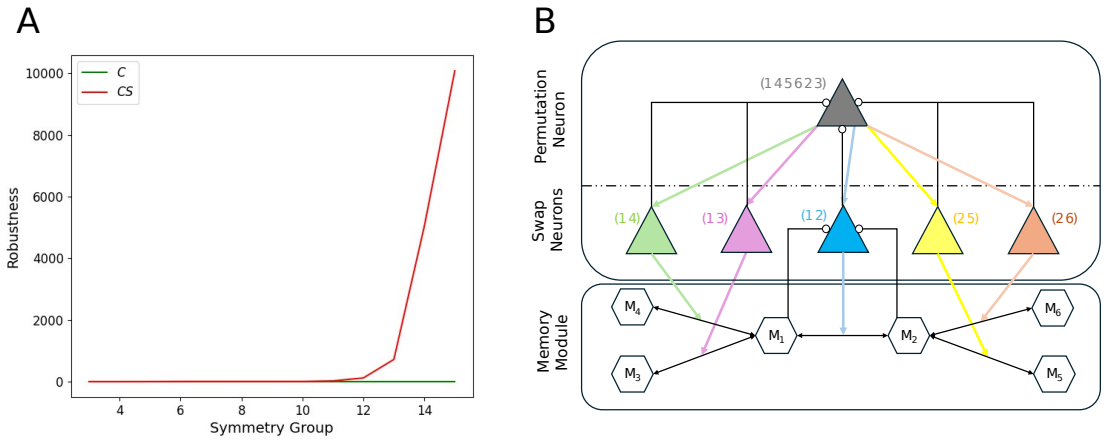

Figure 5: (A) Robustness ($r(G, n)$) of generating sets C and CS for $3 \leq n \leq 15$. (B) A simplified circuit diagram of the network for producing 6-item permutations. All elements depicted in Fig. 2B (such as conjugate neurons, the motor module, mutual inhibition between control neurons) are included in the network but some are left out of the diagram for visual simplicity. Similarly, inhibitory feedback from the memory motifs to swap neurons is only partially depicted. Notice that the memory module is arranged with the community structured architecture highlighted in Fig. 4A.

let $G_{\setminus i}$ denote the remaining swaps in the generating set $G \subset S_n$ after removing all swaps involving the memory motif M_i. For example, given a generating set $G = \{(1\ 2), (2\ 3)\}$, after removing swaps involving M_3, the remaining subset $G_{\setminus 3} = \{(1\ 2)\}$. Consequently, the dysfunction of motif M_i does not impair the swaps contained in $G_{\setminus i}$, nor the permutations composed from swaps in $G_{\setminus i}$. We define $r_i(G)$ as the number of permutations generated by $G_{\setminus i}$, i.e., the number of nodes in $C(S_n, G_{\setminus i})$ connected to $()$. Therefore, the robustness of the entire network is defined as the worst-case value of $r_i(G)$ across all motifs:

$$r(G, n) = \min_i r_i(G), \quad i \in \{1, ...n\}. \tag{4}$$

Fig. 5A visualizes $r(C, n)$ and $r(CS, n)$, showing that the community structured memory module is substantially more robust than the centrally connected memory module. Although the computational cost of the community structured memory module is slightly higher than the centralized architecture (Fig. 4C), we conclude that the community structure achieves a better balance between circuit architecture complexity, computational efficiency, and robustness.

As a proof of concept for our community structured memory module (Fig. 5B), we expand the circuit so that it could produce the 6-item permutation (1 4 5 6 2 3) with neuronal activities shown in Fig. A1. We emphasize that this extension of the circuit required only the addition of control neurons and memory/temporary motifs, while the underlying dynamics of the basic components of the model did not need to be changed, demonstrating the scalability of the circuit.

4. Conclusion and Discussion

The present study investigates the architecture optimization of neural circuit models implementing working memory sequence permutations. Our circuit model is based on a recent mental programming circuit that realizes two-item swapping and three-item permutations Zuo et al. (2025), and we scale the circuit model to longer sequence permutations utilizing the Cayley graph representation of permutation groups. Importantly, we demonstrate that the Cayley graph structure has one-to-one correspondence with the circuit architecture: the basic swaps in the group generating set of the Cayley graph directly specify the connectivity between memory motifs and the associated memory module structure. Since a permutation group can have multiple Cayley graphs with various generating sets, there are multiple circuit architectures implementing the same permutation group while having a trade-off between computational efficiency and circuit architecture complexity. Based on the Cayley graph representation of the circuit architecture, we numerically estimate the computational efficiency, circuit architecture complexity, and the robustness to local failures. Our results favor a community-structured architecture for the memory module, where a few highly connected memory motifs act as relay motifs for exchanging working memory item information.

Due to the working memory capacity limit (usually 4 to 7 items), we only examined circuit models that can permute relatively small sequences (< 9 items), which can be scaled to longer sequences with an architecture optimized by using the Cayley graph. With longer sequences, the community-structured memory module will gradually converge into a **scale-free** network, where the probability of a motif having k connections is given by the power law (Fig. A2B):

$$P(k) \propto k^{-\gamma} \tag{5}$$

This degree distribution implies that the memory module will feature a few highly connected memory motifs to efficiently relay items. Around these highly connected relay motifs, the scale-free memory module would be locally isomorphic to the centralized memory module previously examined (Fig. A2A). Apart from the computational efficiency, scale-free connection topology has been found in cortical circuits He (2014); Li et al. (2010), supporting its biological plausibility.

Our work provides overarching connections of working memory circuit, sequence manipulation, and permutation groups and Cayley graphs. It not only provides theoretical and mechanistic insights on neural circuit mechanism of working memory sequence manipulation, but also the circuit model can be a candidate of future building block for machine learning models in structured sequential tasks such as language processing.

References

R Ben-Yishai, R Lev Bar-Or, and H Sompolinsky. Theory of orientation tuning in visual cortex. *Proceedings of the National Academy of Sciences*, 92(9):3844–3848, 1995.

Albert Compte, Nicolas Brunel, Patricia S Goldman-Rakic, and Xiao-Jing Wang. Synaptic mechanisms and network dynamics underlying spatial working memory in a cortical network model. *Cerebral cortex*, 10(9):910–923, 2000.

Nelson Cowan. The magical number 4 in short-term memory: A reconsideration of mental storage capacity. *Behavioral and brain sciences*, 24(1):87–114, 2001.

Sophie Deneve, Peter E Latham, and Alexandre Pouget. Reading population codes: a neural implementation of ideal observers. *Nature Neuroscience*, 2(8):740–745, 1999.

Joaquin M Fuster and Garrett E Alexander. Neuron activity related to short-term memory. *Science*, 173(3997):652–654, 1971.

John Guckenheimer and Yuri A Kuznetsov. Cusp bifurcation. *Scholarpedia*, 2(4):1852, 2007.

Biyu J He. Scale-free brain activity: past, present, and future. *Trends in cognitive sciences*, 18(9):480–487, 2014.

Mikail Khona and Ila R. Fiete. Attractor and integrator networks in the brain. *Nature Reviews Neuroscience*, 23(12):744–766, December 2022. ISSN 1471-0048. doi: 10.1038/s41583-022-00642-0.

James J Knierim and Kechen Zhang. Attractor dynamics of spatially correlated neural activity in the limbic system. *Annual review of neuroscience*, 35:267–285, 2012.

Xiaoli Li, Gaoxiang Ouyang, Astushi Usami, Yuji Ikegaya, and Attila Sik. Scale-free topology of the ca3 hippocampal network: a novel method to analyze functional neuronal assemblies. *Biophysical journal*, 98(9):1733–1741, 2010.

Steven J Luck and Edward K Vogel. The capacity of visual working memory for features and conjunctions. *Nature*, 390(6657):279–281, 1997.

Earl K Miller and Jonathan D Cohen. An integrative theory of prefrontal cortex function. *Annual review of neuroscience*, 24(1):167–202, 2001.

Earl K Miller, Mikael Lundqvist, and André M Bastos. Working memory 2.0. *Neuron*, 100 (2):463–475, 2018.

MA Stephens. Random walk on a circle. *Biometrika*, 50(3/4):385–390, 1963.

Michael J Tarr and Steven Pinker. Mental rotation and orientation-dependence in shape recognition. *Cognitive psychology*, 21(2):233–282, 1989.

The Sage Developers. *SageMath, the Sage Mathematics Software System*, 2022. URL https://www.sagemath.org. DOI 10.5281/zenodo.6259615.

Zhenghe Tian, Jingwen Chen, Cong Zhang, Bin Min, Bo Xu, and Liping Wang. Mental programming of spatial sequences in working memory in the macaque frontal cortex. *Science*, 385(6716):eadp6091, 2024.

Klaus Wimmer, Duane Q Nykamp, Christos Constantinidis, and Albert Compte. Bump attractor dynamics in prefrontal cortex explains behavioral precision in spatial working memory. *Nature neuroscience*, 17(3):431, 2014.

Si Wu, Kosuke Hamaguchi, and Shun-ichi Amari. Dynamics and computation of continuous attractors. *Neural Computation*, 20(4):994–1025, 2008.

Si Wu, KY Michael Wong, CC Alan Fung, Yuanyuan Mi, and Wenhao Zhang. Continuous attractor neural networks: candidate of a canonical model for neural information representation. *F1000Research*, 5, 2016.

Junfeng Zuo, Cheng Xue, Si Wu, and Wenhao Zhang. Towards a mental programming neural circuit: Insights from working memory sequence manipulation. *bioRxiv*, pages 2025–07, 2025.

Appendix A. Dynamics of Circuit Components

This section provides a mathematical description of the dynamics of components in the swapping circuit described in 2. This information is adapted from Zuo et al. (2025).

A.1. E neurons' dynamics

E neurons are selective for a 1D angular feature $z \in (-\pi, \pi]$. Denote θ_j as the preferred stimulus feature of the j-th E neuron, and the preferred feature of all N_E E neurons, $\{\theta_j\}_{j=1}^{N_E}$, uniformly cover the whole space z (Fig. 3C). Mathematically, in the continuum limit of an infinite number of neurons ($\theta_j \to \theta$), the dynamics of E neurons can be written as (Deneve et al., 1999; Wu et al., 2008),

$$\tau \dot{\mathbf{u}}_m^E(\theta, t) = -\mathbf{u}_m^E(\theta, t) + \rho \sum_n (\mathbf{W}_{mn} * \mathbf{r}_n^E)(\theta, t), \quad (m, n = \{M_1, M_2, T_{12}, T_{21}\}) \tag{6}$$

where $\mathbf{u}_m^E(\theta, t)$ and $\mathbf{r}_m^E(\theta, t)$ represent, respectively, the synaptic inputs and firing rates of neurons preferring $z = \theta$. m is the index of memory motifs that is referred to as one of the motifs in Fig. 3B. τ is the time constant, and $\rho = N_E/2\pi$ is the neuronal density covering the stimulus feature space.

A.2. Recurrent connection kernel.

$\mathbf{W}_{mn}^{EE}(\theta)$ is the recurrent connection kernel from E neurons in memory motif n to the motif m, which are modeled as Gaussian functions in the model (Fig. 3C),

$$\mathbf{W}_{mn}^{EE}(\theta) = w_{mn}^{EE} \left(\sqrt{2\pi} a\right)^{-1} \exp(-\theta^2/2a^2), \tag{7}$$

where w_{mn}^{EE} (scalar) is the peak recurrent weight and to be adjusted for realizing mental programming. a the connection width across the stimulus feature space. The symbol $*$ denotes the convolution, i.e., $\mathbf{W}(\theta) * \mathbf{r}(\theta) = \int \mathbf{W}(\theta - \theta')\mathbf{r}(\theta')d\theta'$. The memory motifs form a circle structure in connection topology ($M_1 \to T_{12} \to M_2 \to T_{21} \to M_1$), with the connection weight shown in Fig. 3B and D.

A.3. Divisive normalization and gain modulation of E neurons.

Let $\mathbf{r}_m^E(\theta, t)$ denote the firing rate of E neurons in memory motif m.

$$\mathbf{r}_m^E(\theta, t) = g_m(r_c(t)) \cdot [\mathbf{u}_m(\theta, t)]_+^2 / [1 + w^{EI} \cdot r_m^I(t)], \quad \text{with} \quad [x]_+ = \max(x, 0), \tag{8}$$

where $r_m^I(t)$ (scalar) is the instantaneous mean firing rate of I neuron pool (Eq. 9). w^{EI} (a positive scalar) characterizes the effective inhibition strength from I neurons to E neurons.

A.4. Shared inhibition across memory subspaces.

Denote $r_m^I(t)$ (scalar) to be the mean firing rate of inhibitory neurons in memory motif m.

$$\tau \dot{r}_m^I(t) = -r_m^I(t) + \sum_n w_{mn}^{IE} \rho \int [\mathbf{u}_n^E(\theta', t)]_+^2 d\theta', \tag{9}$$

where w_{mn}^{IE} (scalar) is the weight from E neurons in motif n to the I neuron pool in motif m.

A.5. Swapping Control Sub-Module Dynamics

Denote u_c and r_c as synaptic input and firing rate of E neurons in the control circuit respectively,

$$\tau \dot{u}_c(t) = -u_c(t) + w_c \cdot r_c(t) + I_{\text{rule}} - I_{CM}, \quad r_c(t) = g_c(t) \cdot [u_c(t)]_+^2 / \big(1 + k_c [u_c(t)]_+^2\big). \quad (10)$$

I_{rule} is the external rule signal, and I_{CM} is the inhibitory feedback from memory subspaces. The control circuit modulates the gain of E neurons in the temporary motif,

$$g_m(t) = W_{TC} \cdot r_c(t), \quad (m = \{T_{12}, T_{21}\}) \quad (11)$$

where W_{TC} denotes the weight from swapping control module to temporary WM motifs.

Appendix B. Constructing Cayley Graphs and Generating Sets

All Cayley graphs and subgroups of S_n were constructed using the Sage math library (The Sage Developers, 2022), which was also used to compute the distances on the Cayley graphs to the identity node. The community structured memory module in Sec. 3 is generated by recursively applying a single rule: to each motif u in the memory, attach $\deg(u) + 1$ additional motifs, where $\deg(u)$ is the number of motifs connected to u. The generating set CS of swaps implemented by this memory module is simply the set of motif connections. For the purposes of comparing computational efficiency, we rewrite this as an iterative algorithm so that the network contains only n memory motifs (1).

Algorithm 1: Community Structured Generating Set

Given $n \geq 2$:

1. Initialize graph G with vertices 1 and 2 and edge $\{1, 2\}$
2. While $|\text{vertices}(G)| < n$:
   (a) Let $C \leftarrow G$
   (b) For each $u \in \text{vertices}(C)$:
      i. Repeat $(\deg_C(u) + 1)$ times:
         A. Add new vertex $v \leftarrow |\text{vertices}(G)| + 1$ to G
         B. Add edge $\{u, v\}$ to G
         C. If $|\text{vertices}(G)| = n$ return edges(G)

Appendix C. 6-item Circuit Simulation Data

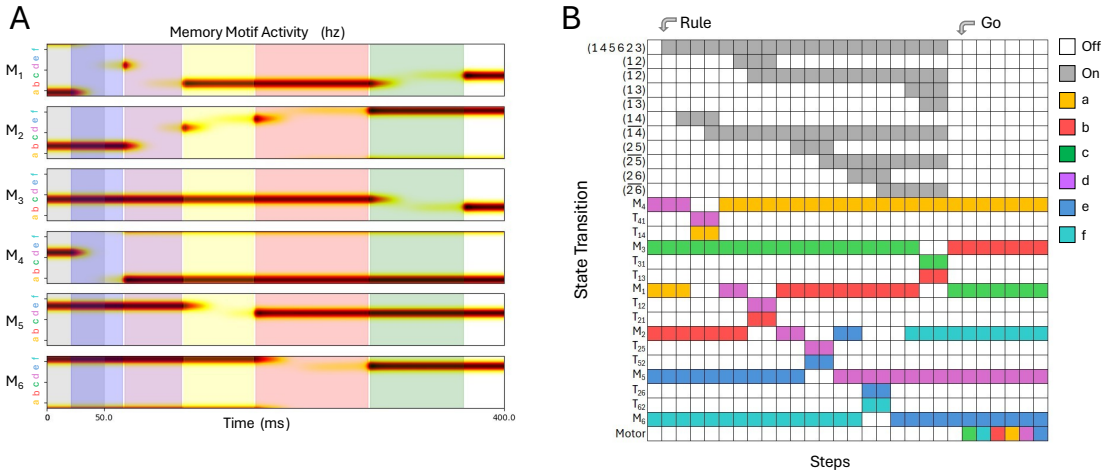

Figure A1: (A) Memory motif activity for 6-item permutation circuit. (B) State transition matrix summarizing the activity of circuit components.

Appendix D. Degree Distribution of CS Memory Module

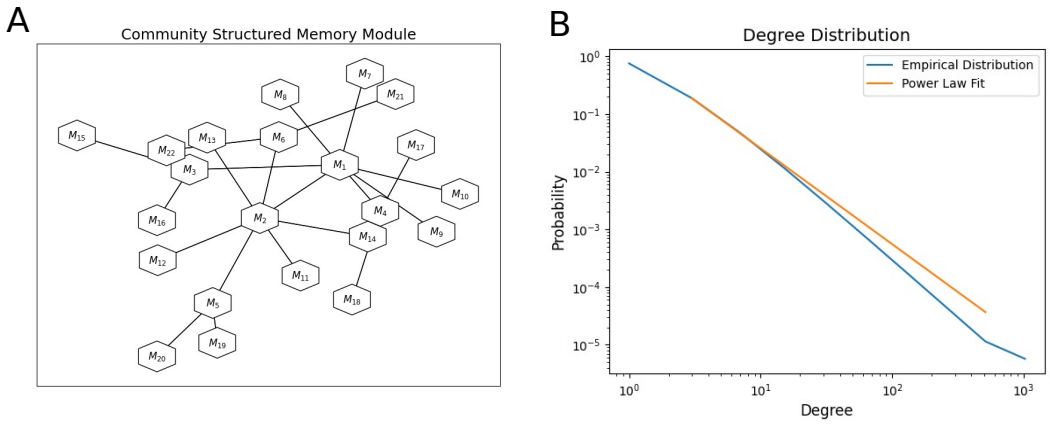

Figure A2: (A) Visualization of the community structured memory module with 22 motifs. (B) Degree distribution of community structured memory module with 349526 motifs. Both the x-axis and y-axis are plotted logarithmically. The blue line plots the observed distribution and the orange line plots a power law curve of best fit. On a log-log scale, a power law distribution appears as a straight line.