Appendix

A. Relationship between OPA and DPA

Lemma 2. *For* $\forall n_i, n_j \in \mathbb{N}^+$,

$$\sum_{i=1}^{n_i}\sum_{j=1}^{n_j} q_j k_{ij} v_i = \sum_{j=1}^{n_j}\sum_{i=1}^{n_i} q_j k_{ij} v_i \tag{15}$$

where $q_j, k_{ij}, v_i \in \mathbb{R}$.

Proof. We will prove by induction for all $n_j \in \mathbb{N}^+$.

Base case: when $n_j = 1$, the $LHS = RHS = \sum_i^{n_i} q_1 k_{i1} v_i$. Let $t \in \mathbb{N}^+$ be given and suppose Eq. 15 is true for $n_j = t$. Then

$$\begin{aligned}\sum_{i=1}^{n_i}\sum_{j=1}^{t+1} q_j k_{ij} v_i &= \sum_{i=1}^{n_i}\left(q_{t+1}k_{it+1}v_i + \sum_{j=1}^{t} q_j k_{ij} v_i\right)\\ &= \sum_{i=1}^{n_i} q_{t+1}k_{it+1}v_i + \sum_{i=1}^{n_i}\sum_{j=1}^{t} q_j k_{ij} v_i\\ &= \sum_{i=1}^{n_i} q_{t+1}k_{it+1}v_i + \sum_{j=1}^{t}\sum_{i=1}^{n_i} q_j k_{ij} v_i\\ &= \sum_{j=1}^{t+1}\sum_{i=1}^{n_i} q_j k_{ij} v_i\end{aligned}$$

Thus, Eq. 15 holds for $n_j = t+1$ and $\forall n_j \in \mathbb{N}^+$ by the principle of induction. □

Proposition 3. *Assume that $\mathcal{S}$ is a linear transformation: $\mathcal{S}(x) = ax + b$ ($a, b, x \in \mathbb{R}$), we can extract A° from $A^{\otimes}$ by using an element-wise linear transformation $\mathcal{F}(x) = a^f \odot x + b^f$ ($a^f, b^f, x \in \mathbb{R}^{d_{qk}}$) and a contraction $\mathcal{P}$: $\mathbb{R}^{d_{qk}\times d_v} \to \mathbb{R}^{d_v}$ such that*

$$A^{\circ}(q,K,V) = \mathcal{P}\left(A^{\otimes}(q,K,V)\right) \tag{16}$$

where

$$A^{\circ}(q,K,V) = \sum_{i=1}^{n_{kv}} \mathcal{S}(q\cdot k_i)\, v_i \tag{17}$$

$$A^{\otimes}(q,K,V) = \sum_{i=1}^{n_{kv}} \mathcal{F}(q\odot k_i)\otimes v_i \tag{18}$$

Proof. We derive the LHS. Let u_i denote the scalar $\mathcal{S}(q\cdot k_i)$, then

$$\begin{aligned}u_i = \mathcal{S}(q\cdot k_i) &= \mathcal{S}\left(\sum_{j=1}^{d_{qk}} q_j k_{ij}\right)\\ &= \sum_{j=1}^{d_{qk}} a q_j k_{ij} + b\end{aligned}$$

where q_j and k_{ij} are the j-th elements of vector q and k_i, respectively. Let $l \in \mathbb{R}^{d_v}$ denote the vector $A^{\circ}(q,K,V) = \sum_{i=1}^{n_{kv}} u_i v_i$, then the t-th element of l is

$$\begin{aligned}l_t &= \sum_{i=1}^{n_{kv}} u_i v_{it}\\ &= \sum_{i=1}^{n_{kv}}\left(\sum_{j=1}^{d_{qk}} a q_j k_{ij} + b\right) v_{it}\\ &= \sum_{i=1}^{n_{kv}}\sum_{j=1}^{d_{qk}} a q_j k_{ij} v_{it} + b\sum_{i=1}^{n_{kv}} v_{it}\\ &= a\sum_{i=1}^{n_{kv}}\sum_{j=1}^{d_{qk}} q_j k_{ij} v_{it} + b\sum_{i=1}^{n_{kv}} v_{it}\end{aligned} \tag{19}$$

We derive the RHS. Let d_i denote the vector $\mathcal{F}(q\odot k_i)$, then the j-th element of d_i is

$$\begin{aligned}d_{ij} &= \mathcal{F}(q_j k_{ij})\\ &= a_j^f q_j k_{ij} + b_j^f\end{aligned} \tag{20}$$

Let $e \in \mathbb{R}^{d_{qk}\times d_v}$ denote the matrix $A^{\otimes}(q,K,V) = \sum_{i=1}^{n_{kv}} d_i \otimes v_i$, then the j-th row, t-column element of e is

$$\begin{aligned}e_{jt} &= \sum_{i=1}^{n_{kv}} d_{ij} v_{it}\\ &= \sum_{i=1}^{n_{kv}}\left(a_j^f q_j k_{ij} + b_j^f\right) v_{it}\\ &= \sum_{i=1}^{n_{kv}} a_j^f q_j k_{ij} v_{it} + b_j^f \sum_{i=1}^{n_{kv}} v_{it}\end{aligned} \tag{21}$$

Let $r \in \mathbb{R}^{d_v}$ denote the vector $\sum_{j=1}^{d_{qk}} e_j$, then the t-th element of r is

Model	Addition complexity	Multiplication complexity	Physical storage for relationships
DPA	$O\left(\left(d_{qk}n_q+d_v\right)n_{kv}\right)$	$O\left(\left(d_{qk}+d_v\right)n_qn_{kv}\right)$	$O\left(n_qn_{kv}\right)$
OPA	$O\left(n_qn_{kv}d_{qk}d_v\right)$	$O\left(n_qd_{qk}d_v\right)$	$O\left(n_qd_{qk}d_v\right)$

Table 5. Computational complexity of DPA and OPA with n_q queries and n_{kv} key-value pairs. d_{qk} denotes query or key size, while d_v value size.

Model	Wall-clock time (second)
LSTM	0.1
NTM	1.8
RMC	0.3
STM	0.3

Table 6. Wall-clock time to process a batch of data on Priority Sort task. The batch size is 128. All models are implemented using Pytorch, have around 1 million parameters and run on the same machine with Tesla V100-SXM2 GPU.

$$\begin{aligned}r_t&=\sum_{j=1}^{d_{qk}}e_{jt}\\&=\sum_{j=1}^{d_{qk}}\left(\sum_{i=1}^{n_{kv}}a_j^fq_jk_{ij}v_{it}+b_j^f\sum_{i=1}^{n_{kv}}v_{it}\right)\\&=\sum_{j=1}^{d_{qk}}\sum_{i=1}^{n_{kv}}a_j^fq_jk_{ij}v_{it}+\sum_{j=1}^{d_{qk}}b_j^f\sum_{i=1}^{n_{kv}}v_{it}\end{aligned}\tag{22}$$

We can always choose $a_j^f=a$ and $\sum_{j=1}^{d_{qk}}b_j^f=b$. Eq. 22 becomes,

$$r_t=a\sum_{j=1}^{d_{qk}}\sum_{i=1}^{n_{kv}}q_jk_{ij}v_{it}+b\sum_{i}^{n_{kv}}v_{it}$$

According to Lemma 2, $l_t=r_t\ \forall d_{qk},n_{kv}\in\mathbb{N}^+\Rightarrow l=r$. Also, $\exists\mathcal{P}$ as a contraction: $\mathcal{P}\left(X\right)=a_pX$ with $a_p=\left[1,...,1\right]\in\mathbb{R}^{1\times d_{qk}}$. □

We compare the complexity of DPA and OPA in Table 5. In general, compared to that of DPA, OPA's complexity is increased by an order of magnitude, which is equivalent to the size of the patterns. In practice, we keep that value small (96) to make the training efficient. That said, due to its high-order nature, our memory model still maintains enormous memory space. In terms of speed, STM's running time is almost the same as RMC's and much faster than that of DNC or NTM. Table 6 compares the real running time of several memory-based models on Priority Sort task.

B. Relationship between OPA and bi-linear model

Proposition 4. *Given the number of key-value pairs $n_{kv}=1$, and $\mathcal{G}$ is a high dimensional linear transformation $\mathcal{G}:\mathbb{R}^{d_{qk}\times d_v}\rightarrow\mathbb{R}^n$, $\mathcal{G}\left(X\right)=W^g\mathcal{V}\left(X\right)$ where $W^g\in\mathbb{R}^{n\times d_{qk}d_v}$, $\mathcal{V}$ is a function that flattens its input tensor, then $\mathcal{G}\left(A^{\otimes}\left(q,K,V\right)\right)$ can be interpreted as a bi-linear model between f and v_1, that is*

$$\mathcal{G}\left(A^{\otimes}\left(q,K,V\right)\right)\left[s\right]=\sum_{j=1}^{d_{qk}}\sum_{t=1}^{d_v}W^g\left[s,j,t\right]f\left[j\right]v_1\left[t\right]\tag{23}$$

where $W^g\left[s,j,t\right]=W^g\left[s\right]\left[\left(j-1\right)d_v+t\right]$, $s=1,...,n$, $j=1,...,d_{qk}$, $t=1,...,d_v$, and $f=\mathcal{F}\left(q\odot k_1\right)$.

Proof. By definition,

$$\begin{aligned}\mathcal{V}\left(\mathcal{F}\left(q\odot k_1\right)\otimes v_1\right)\left[\left(j-1\right)d_v+t\right]&=\left(\mathcal{F}\left(q\odot k_1\right)\otimes v_1\right)\left[j\right]\left[t\right]\\&=\mathcal{F}\left(q\odot k_1\right)\left[j\right]v_1\left[t\right]\end{aligned}$$

We derive the LHS,

$$\begin{aligned}\mathcal{G}\left(A^{\otimes}\left(q,K,V\right)\right)\left[s\right]&=\left(W^g\mathcal{V}\left(\mathcal{F}\left(q\odot k_1\right)\otimes v_1\right)\right)\left[s\right]\\&=\sum_{u=1}^{d_{qk}d_v}W^g\left[s\right]\left[u\right]\mathcal{V}\left(\mathcal{F}\left(q\odot k_1\right)\otimes v_1\right)\left[u\right]\\&=\sum_{\left(j-1\right)d_v+t}^{d_{qk}d_v}\left(W^g\left[s\right]\left[\left(j-1\right)d_v+t\right]\right.\\&\quad\left.\times\mathcal{V}\left(\mathcal{F}\left(q\odot k_1\right)\otimes v_1\right)\left[\left(j-1\right)d_v+t\right]\right)\\&=\sum_{j=1}^{d_{qk}}\sum_{t=1}^{d_v}W^g\left[s,j,t\right]\mathcal{F}\left(q\odot k_1\right)\left[j\right]v_1\left[t\right]\end{aligned}$$

which equals the RHS. □

Prop. 4 is useful since it demonstrates the representational capacity of OPA is at least equivalent to bi-linear pooling, which is richer than low-rank bi-linear pooling using Hadamard product, or bi-linear pooling using identity matrix of the bi-linear form (dot product), or the vanilla linear models using traditional neural networks.

Proposition 5. *Given the number of queries $n_q=d_{qk}$, the number of key-value pairs $n_{kv}=1$, $\mathcal{M}_t^r=\mathrm{SAM}_\theta\left(M\right)$ where M is an instance of the item memory in the past, and $\mathcal{G}$ is a high dimensional linear transformation $\mathcal{G}$:*

$\mathbb{R}^{n_q\times d_{qk}\times d_v}\to\mathbb{R}^{d_{qk}\times d_v}$, *$\mathcal{G}(X)=W^g\mathcal{V}_f(X)$ where $W^g\in\mathbb{R}^{d_{qk}\times n_q d_{qk}}$, $\mathcal{V}_f$ is a function that flattens the first two dimensions of its input tensor, then Eq. 13 can be interpreted as a Hebbian update to the item memory.*

Proof. Let $k_1=M_k$ and $v_1=M_v$ when $n_{kv}=1$, by definition $\mathcal{V}_f\left(\text{SAM}_\theta(M)\right)[(s-1)d_{qk}+j,t]=\mathcal{F}\left(M_q[s]\odot k_1\right)[j]\,v_1[t]$. We derive,

$$\begin{aligned}\mathcal{G}\left(\text{SAM}_\theta(M)\right)[i,t]&=\left(W^g\mathcal{V}_f\left(\text{SAM}_\theta(M)\right)\right)[i,t]\\&=\sum_{u=1}^{n_q d_{qk}}W^g[i,u]\,\mathcal{V}_f\left(\text{SAM}_\theta(M)\right)[u,t]\\&=\sum_{(s-1)d_{qk}+j=1}^{n_q d_{qk}}\left(W^g[i,(s-1)d_{qk}+j]\right.\\&\quad\left.\times\mathcal{F}\left(M_q[s]\odot k_1\right)[j]\,v_1[t]\right)\\&=\sum_{s=1}^{n_q}\sum_{j=1}^{d_{qk}}W^g[i,s,j]\,f[s,j]\,v_1[t]\end{aligned}\tag{24}$$

where $f[s,j]=\mathcal{F}\left(M_q[s]\odot k_1\right)[j]=\mathcal{F}\left(M_q[s,j]\,k_1[j]\right)$. It should be noted that with trivial rank-one W^g: $W^g[i]=d_i\mathcal{V}_f(I)$, $d_i\in\mathbb{R}$, I is the identity matrix, Eq. 24 becomes

$$\begin{aligned}&\mathcal{G}\left(\text{SAM}_\theta(M)\right)[i,t]=d[i]\,v_1[t]\\&\Rightarrow\mathcal{G}\left(\text{SAM}_\theta(M)\right)=d\otimes v_1\end{aligned}$$

where $d\in\mathbb{R}^{d_{qk}}, d[i]=d_i\sum_{s=1}^{n_q}\mathcal{F}\left(M_q[s,s]\,k_1[s]\right)$. Eq. 13 reads

$$\mathcal{M}_t^i=\mathcal{M}_t^i+\alpha_3 d\otimes v_1$$

which is a Hebbian update with the updated value v_1. As v_1 is a stored pattern extracted from M encoded in the relational memory, the item memory is enhanced with a long-term stored value from the relational memory. □

C. OPA and SAM as associative memory[1]

Proposition 6. *If $\mathcal{P}$ is a contraction: $\mathbb{R}^{d_{qk}\times d_v}\to\mathbb{R}^{d_v}$, $\mathcal{P}(X)=a_pX, a_p\in\mathbb{R}^{1\times d_{qk}}$, then $A^{\otimes}(q,K,V)$ is an associative memory that stores patterns $\{v_i\}_{i=1}^{n_{kv}}$ and $\mathcal{P}\left(A^{\otimes}(q,K,V)\right)$ is a retrieval process. Perfect retrieval is possible under the following three conditions,*

(1) $\{k_i\}_{i=1}^{n_{kv}}$ form a set of linearly independent vectors

(2) $q_i\neq 0$, $i=1,...,d_{qk}$

(3) $\mathcal{F}$ is chosen as $\mathcal{F}(x)=a^f\odot x$ ($a^f,x\in\mathbb{R}^{d_{qk}}$, $a_i^f\neq 0$, $i=1,...,d_{qk}$)

Proof. By definition, $A^{\otimes}(q,K,V)$ forms a hetero-associative memory between $x_i=\mathcal{F}(q\odot k_i)$ and v_i. If $\{x_i\}_{i=1}^{n_{kv}}$ are orthogonal, given some $\mathcal{P}$ with $a_p=\frac{x_j^\top}{\|x_j^\top\|}$, then

$$\begin{aligned}\mathcal{P}\left(A^{\otimes}(q,K,V)\right)&=\frac{x_j^\top}{\|x_j^\top\|}\sum_{i=1}^{n_{kv}}x_i\otimes v_i\\&=\sum_{i=1,i\neq j}^{n_{kv}}\frac{\left(x_j^\top x_i\right)}{\|x_j^\top\|}v_i^\top+\frac{\left(x_j^\top x_j\right)}{\|x_j^\top\|}v_j^\top\\&=v_j^\top\end{aligned}$$

Hence, we can perfectly retrieve some stored pattern v_j using its associated $\mathcal{P}$. In practice, linearly independent $\{x_i\}_{i=1}^{n_{kv}}$ is enough for perfect retrieval since we can apply Gram–Schmidt process to construct orthogonal $\{x_i\}_{i=1}^{n_{kv}}$. Another solution is to follow Widrow-Hoff incremental update

$$\begin{aligned}&A^{\otimes}(q,K,V)(0)=0\\&A^{\otimes}(q,K,V)(i)=A^{\otimes}(q,K,V)(i-1)\\&\qquad+\left(v_i-A^{\otimes}(q,K,V)(i-1)\,x_i\right)\otimes x_i\end{aligned}$$

which also results in possible perfect retrieval given $\{x_i\}_{i=1}^{n_{kv}}$ are linearly independent.

Now, we show that if (1) (2) (3) are satisfied, $\{x_i\}_{i=1}^{n_{kv}}$ are linearly independent using proof by contradiction. Assume that $\{x_i\}_{i=1}^{n_{kv}}$ are linearly dependent, $\exists\{\alpha_i\in\mathbb{R}\}_{i=1}^{n_{kv}}$, not all zeros such that

$$\begin{aligned}\overrightarrow{0}&=\sum_{i=1}^{n_{kv}}\alpha_i x_i=\sum_{i=1}^{n_{kv}}\alpha_i\mathcal{F}(q\odot k_i)\\&=\sum_{i=1}^{n_{kv}}\alpha_i\left(a^f\odot(q\odot k_i)\right)\\&=\left(a^f\odot q\right)\odot\left(\sum_{i=1}^{n_{kv}}\alpha_i k_i\right)\end{aligned}\tag{25}$$

As (2) (3) hold true, Eq. 25 is equivalent to

$$\overrightarrow{0}=\sum_{i=1}^{n_{kv}}\alpha_i k_i$$

which contradicts (1). □

[1]In this section, we use these following properties without explanation: $a^\top(b\otimes c)=\left(a^\top b\right)c^\top$ and $(b\otimes c)\,a=\left(c^\top a\right)b$.

Prop. 6 is useful as it points out the potential of our OPA formulation for accurate associative retrieval over several key-value pairs. That is, despite that many items are extracted to form the relational representation, we have the chance to reconstruct any items perfectly if the task requires item memory. As later we use neural networks to generate k and q, the model can learn to satisfy conditions (1) and (2). Although in practice, we use element-wise $\tanh$ to offer non-linear transformation, which is different from (3), empirical results show that our model still excels at accurate associative retrieval.

Proposition 7. *Assume that the gates in Eq. 10 are kept constant $F_t = I_t = 1$, the item memory construction is simplified to*

$$M = \sum_{i=1}^{N+1} x_i \otimes x_i,$$

where $\{x_i\}_{i=1}^{N+1}$ are positive input patterns after feed-forward neural networks and the relational memory construction is simplified to

$$\mathcal{M}^r = \text{SAM}_\theta(M),$$

and layer normalizations are excluded, then the memory retrieval is a two-step contraction

$$v^r = \text{softmax}\left(z^\top\right)\mathcal{M}^r f(x)$$

Proof. Without loss of generality, after seeing $N+1$ patterns $\{x_i\}_{i=1}^{N+1}$, SAM is given a (noisy or incomplete) query pattern x that corresponds to some stored pattern $x_p = x_{N+1}$, that is

$$\begin{cases} x_p^\top x \approx 1 \\ x_i^\top x \approx 0 \quad i = \overline{1,N} \end{cases}$$

Unrolling Eq. 8 yields

$$\begin{aligned}
\text{SAM}_\theta(M)[s] &= \sum_{j=1}^{n_{kv}} \mathcal{F}\left(M_q[s] \odot M_k[j]\right) \otimes M_v[j] \\
&= \sum_{j=1}^{n_{kv}} \mathcal{F}\left(W_q[s]\left(\sum_{i=1}^{N+1} x_i \otimes x_i\right)\right. \\
&\left.\odot W_k[j]\left(\sum_{i=1}^{N+1} x_i \otimes x_i\right)\right) \\
&\otimes W_v[j]\left(\sum_{i=1}^{N+1} x_i \otimes x_i\right) \\
&= \sum_{j=1}^{n_{kv}} \mathcal{F}\left(\left(\sum_{i=1}^{N} W_q[s] x_i \otimes x_i\right.\right. \\
&\left. + W_q[s] x_p \otimes x_p\right) \\
&\left.\odot\left(\sum_{i=1}^{N} W_k[j] x_i \otimes x_i + W_k[j] x_p \otimes x_p\right)\right) \\
&\otimes\left(\sum_{i=1}^{N} W_v[j] x_i \otimes x_i + W_v[j] x_p \otimes x_p\right)
\end{aligned} \tag{26}$$

When $d > N$, it is generally possible to find W_q, W_k and W_v that satisfy the following system of equations:

$$\begin{cases}
W_q[s] x_i &= 0, i = \overline{1,N}, \\
W_q[s] x_p &= 1 \\
W_k[j] x_i &= 0, i = \overline{1,N} \\
W_k[j] x_p &= 1 \\
W_v[j] x_i &= 1, i = \overline{1,N} \\
W_v[j] x_p &= 1
\end{cases}$$

We also assume that $\mathcal{F}$ is chosen as square root function, then Eq. 26 simplifies to

$$\begin{aligned}
\text{SAM}_\theta(M)[s] &= \sum_{j=1}^{n_{kv}} \mathcal{F}\left(x_p \odot x_p\right) \otimes \sum_{i=1}^{N+1} x_i \\
&= n_{kv} x_p \otimes \sum_{i=1}^{N+1} x_i \\
&= n_{kv} \sum_{i=1}^{N+1} x_p \otimes x_i
\end{aligned}$$

The first contraction $\text{softmax}\left(z^\top\right)\mathcal{M}^r$ can be interpreted as an attention to $\{\text{SAM}_\theta(M)[s]\}_{s=1}^{n_q}$, which equals

$$n_{kv} \sum_{i=1}^{N+1} x_p \otimes x_i$$

The second contraction is similar to a normal associative memory retrieval. When we choose $f(x) = \frac{x}{n_{kv}}$, the retrieval reads

$$\begin{aligned} v^r &= \left(n_{kv}\sum_{i=1}^{N+1} x_p \otimes x_i\right)\frac{x}{n_{kv}} \\ &= \sum_{i=1}^{N+1}\left(x_i^\top x\right)x_p \\ &\approx x_p \end{aligned}$$

□

D. Implementation of gate functions

$$F_t\left(\mathcal{M}_{t-1}^i, x_t\right) = W_F x_t + U_F \tanh\left(\mathcal{M}_{t-1}^i\right) + b_F$$

$$I_t\left(\mathcal{M}_{t-1}^i, x_t\right) = W_I x_t + U_I \tanh\left(\mathcal{M}_{t-1}^i\right) + b_I$$

Here, W_F, U_F, W_I, $W_I \in \mathbb{R}^{d\times d}$ are parametric weights, $b_F, b_I \in \mathbb{R}$ are biases and $+$ is broadcasted if needed.

E. Learning curves on ablation study

We plot the learning curves of evaluated modes for Associative retrieval with length 30, 50 and N^{th}-farthest in Fig. 4. For N^{th}-farthest, the last input in the sequence is treated as the query for TPR. We keep the standard number of entities/roles and tune TPR[2] with different hidden dimensions (40, 128, 256) and optimizers (Nadam and Adam). All configurations fail to converge for the normal N^{th}-farthest as shown in Fig. 4 (right). When we reduce the problem size to 4 8-dimensional input vectors, TPR can reach perfect performance, which indicates the problem here is more about scaling to bigger relational reasoning contexts.

F. Implementation of baselines for algorithmic and geometric/graph tasks

Following Graves et al. (2014), we use RMSprop optimizer with a learning rate of 10^{-4} and a batch size of 128 for all baselines.

- LSTM and ALSTM: Both use 512-dimensional hidden vectors for all tasks.
- NTM[3], DNC[4]: Both use a 256-dimensional LSTM controller for all tasks. For algorithmic tasks, NTM uses a 128-slot external memory, each slot is a 32-dimensional vector. Following the standard setting, NTM uses 1 control head for Copy, RAR and 5 control heads for Priority sort. For geometric/graph tasks, DNC is equipped with 64-dimensional 20-slot external memory and 4-head controller. In geometric/graph problems, 20 slots are about the number of points/nodes. We also tested with layer-normalized DNC without temporal link matrix and got similar results.
- RMC[5]: We use the default setting with total 1024 dimensions for memory of 8 heads and 8 slots. We also tried with different numbers of slots $\{1, 4, 16\}$ and Adam optimizer but the performance did not change.
- STM: We use the same setting across tasks $n_q = 8$, $d = 96$, $n_r = 96$. α_1,α_2, and α_3 are learnable.

[2]https://github.com/ischlag/TPR-RNN
[3]https://github.com/vlgiitr/ntm-pytorch
[4]https://github.com/deepmind/dnc

G. Order of relationship

In this paper, we do not formally define the concept of order of relationship. Rather, we describe it using concrete examples. When a problem requires to compute the relationship between items, we regard it as a first-order relational problem. For example, sorting is first-order relational. Copy is even zero-order relational since it can be solved without considering item relationships. When a problem requires to compute the relationship between relationships of items, we regard it as a second-order relational problem and so on.

From this observation, we hypothesize that the computational complexity of a problem roughly corresponds to the order of relationship in the problem. For example, if a problem requires a solution whose computational complexity between $O(N)$ and $O(N^2)$ where N is the input size, it means the solution basically computes the relationship between any pair of input items and thus corresponds to first-order relationship. Table 7 summarizes our hypothesis on the order of relationship in some of our problems.

By design, our proposed STM stores a mixture of relationships between items in a relational memory, which approximately corresponds to a maximum of second-order relational capacity. The distillation process in STM transforms the relational memory to the output and thus determines the order of relationship that STM can offer. We can measure the degree that STM involves in relational mining by analyzing the learned weight $\mathcal{G}_2$ of the distillation process. Intuitively, a high-rank transformation $\mathcal{G}_2$ can capture more relational information from the relational memory. Trivial low-rank $\mathcal{G}$ corresponds to item-based retrieval without much relational mining (Prop. 5). The numerical rank of a matrix A is defined as $r(A) = \|A\|_F^2 / \|A\|_2^2$, which relaxes the exact notion of rank (Rudelson & Vershynin, 2007).

We report the numerical rank of learned $\mathcal{G}_2 \in \mathbb{R}^{6144\times 96}$

[5]https://github.com/L0SG/relational-rnn-pytorch

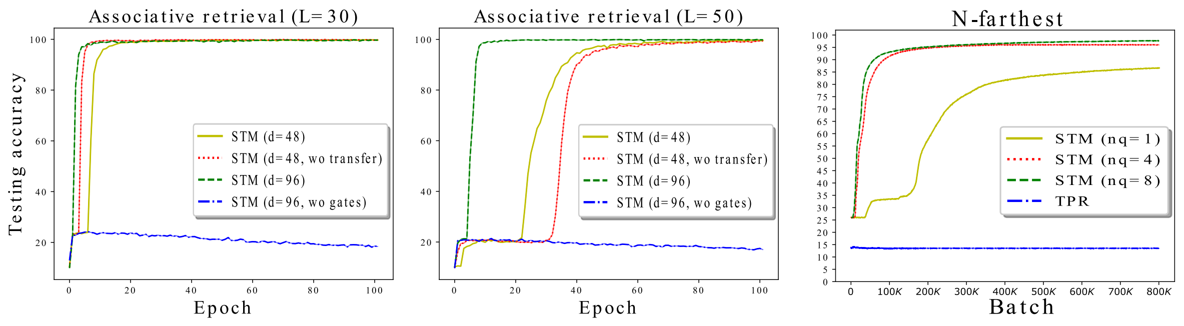

Figure 4. Testing accuracy (%) on associative retrieval L=30 (left), L=50 (middle) and N^{th}-farthest (right).

Task	General complexity	Order
Copy/Associative retrieval	$O(N)$	0
Sort	$O(N \log N)$	1
Convex hull	$O(N \log N)$	1
Shortest path[6]	$O(E \log V)$	1
Minimum spanning tree	$O(E \log V)$	1
RAR/N^{th}-Farthest	$O(N^2 \log N)$	2
Traveling salesman problem	NP-hard	many

Table 7. Order of relationship in some problems.

Task	$r(\mathcal{G}_2)$
Associative retrieval	9.42±0.5
N^{th}-Farthest	83.20±0.2
Copy	79.00±0.3
Sort	79.58±0.1
RAR	83.30±0.2
Convex hull	80.78±0.6
Traveling salesman problem	83.58±0.3
Shortest path	79.81±0.2
Minimum spanning tree	79.57±0.5

Table 8. Mean and std. of numerical rank of the leanred weight $\mathcal{G}_2$ for several tasks. The upper bound for the rank is 96.

for different tasks in Table 8. For each task, we run the training 5 times and take the mean and std. of $r(\mathcal{G}_2)$. The rank is generally higher for tasks that have higher orders of relationship. That said, the model tends to overuse its relational capacity. Even for the zero-order Copy task, the rank for the distillation transformation is still very high.

H. Geometry and graph task description

In this testbed, we use RMSprop optimizer with a learning rate of 10^{-4} and a batch size of 128 for all baselines. STM uses the same setting across tasks $n_q = 8$, $d = 96$, $n_r = 96$.

[6]The input is sequence of triplets, which is equivalent to sequence of edges. Hence, the complexity is based on the number of edges in the graph.

The random one-hot features can be extended to binary features, which is much harder and will be investigated in our future works.

Convex hull Given a set of N points with 2D coordinates, the model is trained to output a list of points that forms a convex hull sorted by coordinates. Training is done with $N \sim [5, 20]$. Testing is done with $N = 5$ and $N = 10$ (no prebuilt dataset available for $N = 20$). The output is a sequence of 20-dimensional one-hot vectors representing the features of the solution points in the convex-hull.

Traveling salesman problem Given a set of N points with 2D coordinates, the model is trained to output a list of points that forms a closed tour sorted by coordinates. Training is done with $N \sim [5, 10]$. Testing is done with $N = 5$ and $N = 10$. The output is a sequence of 20-dimensional one-hot vectors representing the features of the solution points in the optimal tour.

Shortest path The graph is generated according to the following rules: (1) choose the number of nodes $N \sim [5, 20]$, (2) after constructing a path that goes through every node in the graph (to make the graph connected), determine randomly the edge between nodes (number of edges $E \sim [6, 30]$), (3) for each edge set the weight $w \sim [1, 10]$. We generate 100,000 and 10,000 graphs for training and testing, respectively. The representation for an input graph is a sequence of triplets followed by 2 feature vectors representing the source and destination node. The output is a sequence of 40-dimensional one-hot feature vectors representing the solution nodes in the shortest path.

Minimum spanning tree We use the same generated input graphs from the Shortest path task. The representation for an input graph is only a sequence of triplets. The output is a sequence of 40-dimensional one-hot feature vectors representing the features of the nodes in the solution edges of the minimum spanning tree.

Some generated samples of the four tasks are visualized in

Fig. 5. Learning curves are given in Fig. 6.

I. Reinforcement learning task description

We trained Openai Gym's PongNoFrameskip-v4 using Asynchronous Advantage Actor-Critic (A3C) with hyper-parameters: 32 workers, shared Adam optimizer with a learning rate of 10^{-4}, $\gamma = 0.99$. To extract scene features for LSTM and STM, we use 4 convolutional layers (32 kernels with 5×5 kernel sizes and a stride of 1), each of which is followed by a 2×2 max-pooling layer, resulting in 1024-dimensional feature vectors. The LSTM 's hidden size is 512. STM uses $n_q = 8$, $d = 96$, $n_r = 96$.

J. bAbI task description

We use the train/validation/test split introduced in bAbI's en-valid-10k v1.2 dataset. To make STM suitable for question answering task, each story is preprocessed into a sentence-level sequence, which is fed into our STM as the input sequence. The question, which is only 1 sentence, is preprocessed to a query vector. Then, we utilize the Inference module, which takes the query as input to extract the output answer from our relational memory $\mathcal{M}^r$. The preprocessing and the Inference module are the same as in Schlag & Schmidhuber (2018). STM's hyper-parameters are fixed to $n_q = 20$, $d = 90$, $n_r = 96$. We train our model jointly for 20 tasks with a batch size of 128, using Adam optimizer with a learning rate of 0.006, $\beta_1 = 0.9$ and $\beta_2 = 0.99$. Details of all runs are listed in Table 9.

K. Characteristics of memory-based neural networks

Table 10 compares the characteristics of common neural networks with memory. Biological plausibility is determined based on the design of the model. It is unlikely that human memory employs RAM-like behaviors as in NTM, DNC, and RMC. Fixed-size memory is inevitable for online and life-long learning, which also reflects biological plausibility. Relational extraction and recurrent dynamics are often required in powerful models. As shown in the table, our proposed model exhibits all the nice features that a memory model should have.

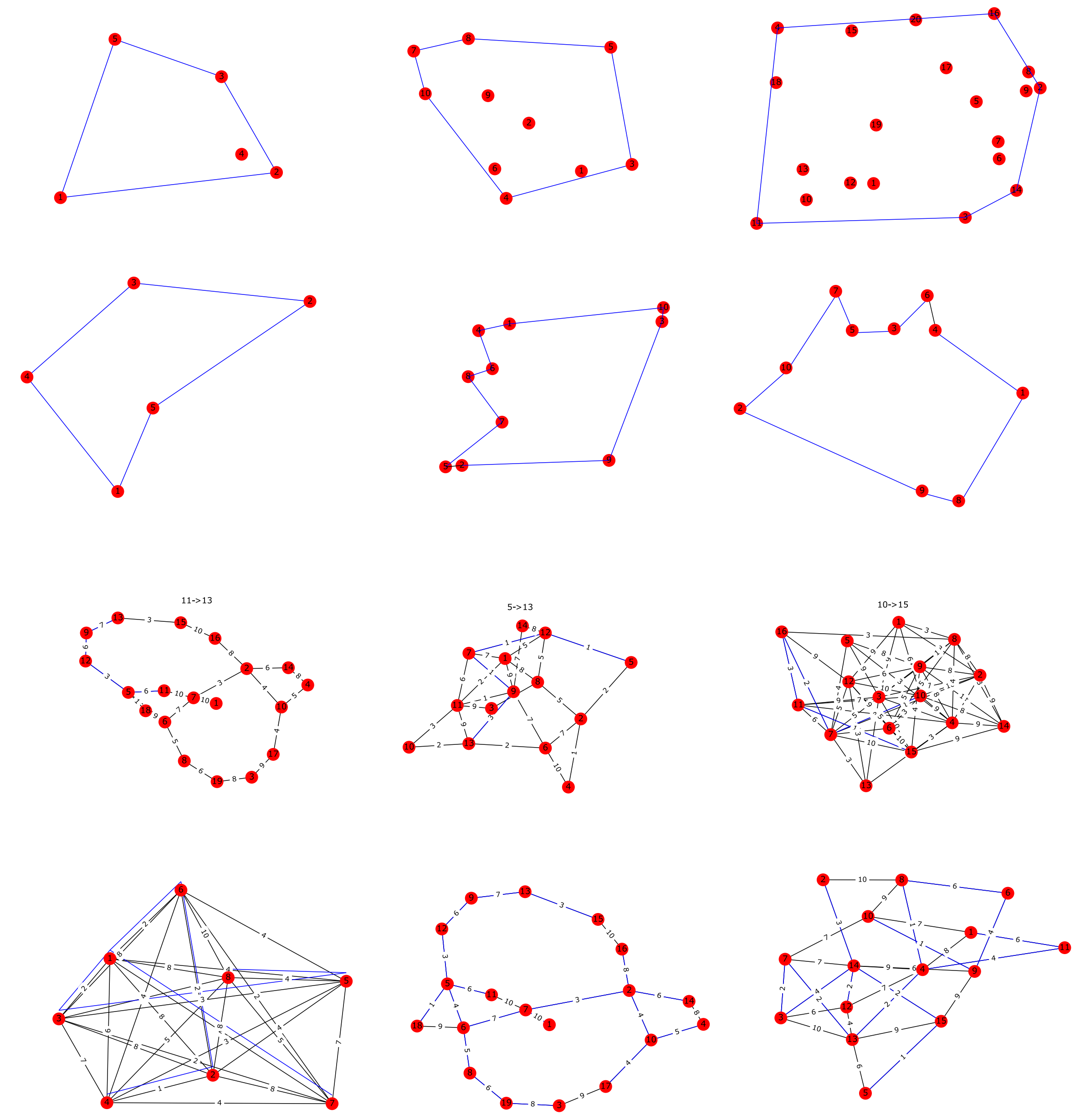

Figure 5. Samples of geometry and graph tasks. From top to bottom: Convex hull, TSP, Shortest path and Minimum spanning tree. Blue denotes the ground-truth solution.

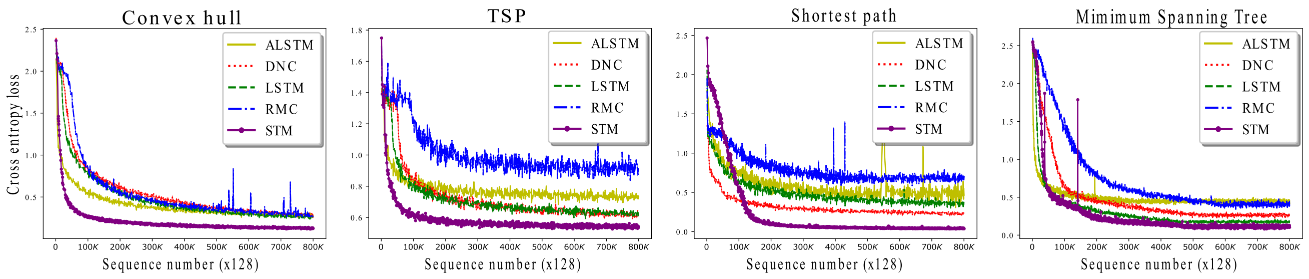

Figure 6. Learning curves on geometry and graph tasks.

Task	run-1	run-2	run-3	run-4	run-5	run-6	run-7	**run-8**	run-9	run-10	Mean
1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.00 ± 0.00
2	0.1	0.6	0.1	0.1	0.7	0.2	0.2	0.0	0.1	0.0	0.21 ± 0.23
3	3.4	3.2	1.0	1.3	2.4	3.8	3.2	0.5	0.9	1.6	2.13 ± 1.14
4	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.00 ± 0.00
5	0.6	0.2	0.6	0.6	0.7	0.5	0.9	0.5	0.7	0.4	0.57 ± 0.18
6	0.0	0.0	0.0	0.0	0.0	0.1	0.0	0.0	0.0	0.0	0.00 ± 0.00
7	1.0	0.9	0.5	0.6	0.9	1.4	1.0	0.6	0.5	0.7	0.81 ± 0.27
8	0.2	0.1	0.1	0.2	0.0	0.0	0.1	0.2	0.1	0.2	0.12 ± 0.07
9	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.00 ± 0.00
10	0.0	0.1	0.0	0.2	0.0	0.0	0.0	0.0	0.0	0.0	0.03 ± 0.06
11	0.0	0.0	0.1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.01 ± 0.03
12	0.1	0.1	0.1	0.0	0.0	0.1	0.0	0.0	0.0	0.0	0.04 ± 0.05
13	0.1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.01 ± 0.03
14	0.1	0.0	0.1	0.0	0.1	0.3	0.0	0.1	0.5	0.4	0.16 ± 0.17
15	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.00 ± 0.00
16	0.3	0.2	0.2	0.3	0.1	0.3	0.6	0.5	0.3	0.1	0.29 ± 0.15
17	0.6	2.6	0.4	0.4	0.5	2.1	3.5	0.5	0.9	0.3	1.18 ± 1.07
18	1.0	0.3	0.2	0.1	0.4	0.4	0.2	0.0	0.1	0.0	0.27 ± 0.28
19	4.4	0.3	0.8	0.0	8.8	0.4	0.3	0.1	4.7	0.8	2.06 ± 2.79
20	0.0	0.0	0.0	0.0	0.0	0.5	0.0	0.0	0.0	0.0	0.00 ± 0.00
Average	0.59	0.43	0.21	0.19	0.73	0.48	0.50	**0.15**	0.44	0.23	0.39 ± 0.18
Failed task (>5%)	0	0	0	0	1	0	0	0	0	0	0.10 ± 0.30

Table 9. Results from 10 runs of STM on bAbI 10k. Bold denotes best run.

Model	Fixed-size memory	Relational extraction	Recurrent dynamics	Biologically plausible
RNN, LSTM	✓	X	✓	✓
NTM, DNC	✓	X	✓	X
RMC	✓	✓	✓	X
Transformer	X	✓	X	X
UT	X	✓	✓	X
Attentional LSTM	X	✓	✓	X
STM	✓	✓	✓	✓

Table 10. Characteristics of some neural memory models