When Do Language Models Need to Be Large?

Zhixun Chen [1] **Yali Du** [1] **David Mguni** [2]

Abstract

Many leading language models (LMs) use high-intensity computational resources both during training and execution. This poses the challenge of lowering resource costs for deployment and faster execution in decision-making tasks among others. We introduce a novel plug & play LM framework named **L**anguage **O**ptimising **N**etwork **Di**stribution (LONDI). LONDI learns to selectively employ large LMs only where complex decision-making and reasoning are required while using low-resource LMs (i.e. LMs require less GPU usage, but may not be able to solve the problem alone) everywhere else. LONDI consists of a system of two (off-)policy networks, an LM, a large LM (LLM), and a reinforcement learning module that uses *switching controls* to quickly learn in which system states to call the LLM. We then introduce a variant of LONDI that maintains budget constraints on LLM calls and hence its resource usage. We test LONDI's performance in a range of tasks in ScienceWorld and BabyAI-Text and demonstrate that LONDI can solve tasks only solvable by resource-intensive LLMs while reducing GPU usage by up to 30%.

1. Introduction

Large language models (LLMs) have emerged as powerful tools that can assist humans to accomplish a wide range of tasks such as medical tasks (Lin et al., 2023c), language education (Caines et al., 2023), autonomous driving (Fu et al., 2023) and recreational games (Feng et al., 2024). As LLMs originate from data center warehouses, they are expected to gradually extend their reach to edge devices such as personal computers, smartphones, and even Internet of Things (IoT) devices (Central, 2023; Qualcomm, 2023). This shift is driven by the desire for enhanced data privacy, availability of AI functionalities, and personalised experiences (Yi et al., 2023). For instance, Qualcomm has effectively implemented stable diffusion, a text-to-image generative LLM model, on smartphones (Qualcomm, 2023). Multimodal LLMs have been integrated into smartphones, enabling precise content searching through natural language queries (Central, 2023).

However, current LLMs require massive computational resources for training and execution, which makes the application of LLMs in edge devices without large memory storage still unrealistic (Ahmed & Wahed, 2020; Yi et al., 2023). Smaller language models (LMs) may offer a practical solution to the computational constraints. Whereas the ability of LLM is significantly correlated with the size of its parameter set (Brown et al., 2020). The performance of smaller LMs is dramatically degraded compared with larger LMs, rendering smaller LMs unable to solve some tasks (Ding et al., 2023). A practical solution is to combine a large LM with a small LM and activate the large LM only under a set of conditions. Though there have been several attempts to combine LMs in this way, the challenge of systematically constructing a set of conditions to activate LMs has not yet been fully resolved.

Dual process theory (DPT) (Kahneman, 2011) posits that human thinking and decision-making involve two separate cognitive processes or systems, with one being characterised as fast, automatic, and intuitive, while the other is described as slow, controlled, and reflective. According to this theory, this delineation enables fast, reactive decision-making in states where complex modes of thought are not required while employing slower yet more sophisticated, complex reasoning where it is required.

Inspired by DPT, we propose a dual language model structure that consists of two LMs and an adaptive reinforcement learning (RL) agent, Switcher as a switch mechanism to automate selective activations of the LMs. By applying a smaller LM (which we call QUICK) as a fast response module and a larger LLM (which we call DEEPTHINK) as the slow thinking module, LONDI's adaptive switch mechanism learns to efficiently manage cooperative delegation between two LMs to solve tasks while reducing the resource cost burden of LMs. Specifically, the switch agent, based on switching control policy (Mguni et al., 2023a;b;c), deter-

[1]King's College, London [2]Queen Mary University, London. Correspondence to: David Mguni <davidmguni@hotmail.com>, Yali Du <yali.du@kcl.ac.uk>.

Published at ICML 2024 Workshop on Foundation Models in the Wild.

mines the states in which to activate DEEPTHINK, while QUICK is utilised in all other states. Therefore, we approach the problem of computational constraints within LLMs from a systematic standpoint using the switching control, which indicates that the switching policy only activates DEEPTHINK at the beneficial set of states. LONDI has a cost parameter c , which plays an important role in calibrating the resource-consumption/performance trade-off of the system of MARL. Larger values of c incur higher costs for each activation of the DEEPTHINK large language model by the Switcher. The in turn makes the Switcher more selective about activating the DEEPTHINK LLM, reserving its activations to a smaller number of states where the boost in performance is greatest. In the limit $c \to \infty$ the Switcher becomes extremely thrifty in which case LONDI solely uses the QUICK model. Moreover, we introduce a variant of LONDI, namely LONDI-B that imposes a budgetary constraint on the number of DEEPTHINK calls. In this setup, LONDI maintains a budget on the number of DEEPTHINK calls allowed.

Overall, LONDI has several advantages:
• By switching to DEEPTHINK only at states where it is beneficial while leveraging the computational thriftiness of QUICK, LONDI solves various tasks while reducing computational expense (see Sec. 3.2, Appendix. E).
• LONDI can preserve fixed budgets on the number of DEEPTHINK calls, balancing performance and computational cost under the limited budgets.(see Sec. 3.2).
• LONDI is a plug-and-play framework that seamlessly adopts any QUICK and DEEPTHINK module. (see Appendix. E).

2. LONDI

2.1. Problem Formulation

To tackle the challenges of computational resources constrains, we introduce an adaptive learner which we call Switcher that decides on the states to activate DEEPTHINK while using the less computationally expensive QUICK language model to determine actions everywhere else. The Switcher needs to make a binary decision (whether to activate DEEPTHINK or not) at each system state, where a state in the current setting is a representation of a specific *context* or *condition*. We can then formalise the Switcher problem as an MDP (described in Appendix A) as the problem involves sequential decision-making (under uncertainty) with Markovian transitions. Specifically, $\mathcal{S}$ represents the system state space, $\boldsymbol{\mathcal{A}} \equiv \mathcal{A}_S \cup \mathcal{A}$ denotes the action set where action is executed in the environment and which is decided by an 'active' language model, $\mathcal{A}_S \equiv \{0, 1\}$ denotes the Switcher's binary action set, $\mathcal{A}$ represents the action set common to both language models, $\mathcal{P}$ indicates the transition after the action and $\mathcal{R} : \mathcal{S} \times \boldsymbol{\mathcal{A}} \to \mathbb{R}$ represents the return of the environment after the action of LLM activated by the Switcher. At any given instant, only one of the DEEPTHINK and QUICK modules is activated and hence able to take action. To make a decision, the Switcher samples a decision g from its policy $\mathfrak{g} : \mathcal{S} \to \{0, 1\}$ where $g = 1$ indicates an activation of the DEEPTHINK module in which case the action $a \sim \pi^{\text{DEEP}}$ is executed while $g = 0$ indicates that no activation of the DEEPTHINK module occurs so the QUICK module is active in which case the action $a \sim \pi^{\text{switch}}$ is executed where π^{QUICK} and π^{DEEP} are policies associated to the QUICK and DEEPTHINK modules respectively.

Under the MDP setting, the goal of the Switcher is to maximise the cumulative return, namely the overall performance of the structure. To prompt Switcher to make selective activations decisions, a fixed cost associated with each activation is imposed on Switcher, represented by a constant value $c < 0$. The incurred costs encourages the Switcher to activate DEEPTHINK model only when the activation is advantageous for the system's performance, either in the current state or in subsequent states. The objective of the Switcher policy $\mathfrak{g}$ is

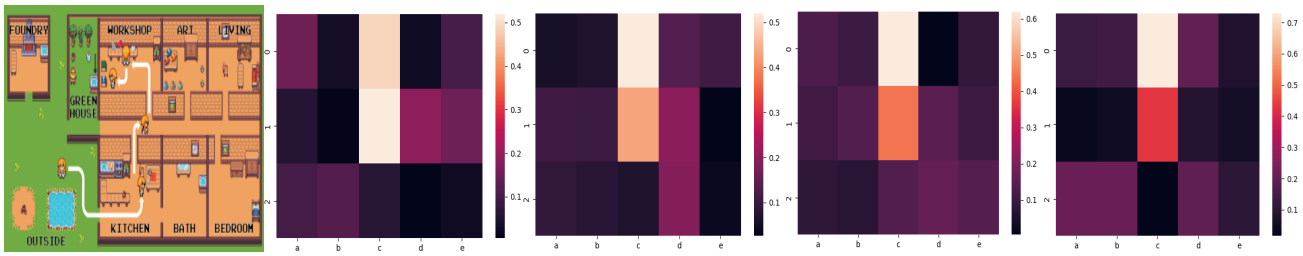
$$v_S(s|\pi, \mathfrak{g}) = \mathbb{E}_{g\sim\mathfrak{g}}\left[\sum_{t=0}^{\infty} \gamma^t \left(r - c \cdot 1(g(s_t))\right) \Big| s_0 = s; a_t \sim \pi\right]$$

and the action-value function of it is $Q_S(s, a|\pi, \mathfrak{g}) = \mathbb{E}_{g\sim\mathfrak{g}}[\sum_{t=0}^{\infty} \gamma^t (r - c \cdot \mathbf{1}(g(s_t)))|s_0 = s; a_0 = a; a_t \sim \pi]$, where π is either π^{QUICK} or π^{DEEP}. With this objective, Switcher's goal is to maximise the system performance by activating DEEPTHINK at the required set of states to enable the task to be solved with the minimal number of DEEPTHINK activation. Therefore, by learning an optimal $\mathfrak{g}$, Switcher acquires the optimal policy for activating DEEPTHINK. For the budget-constrained variant LONDI-B, the activation of DEEPTHINK becomes exorbitantly costly if the number of DEEPTHINK activation surpasses the imposed budget threshold.

2.2. Switching Controls

The Switcher is tasked with learning the set of states that require the additional decision capacity provided by the DEEPTHINK model in order to achieve the optimal policy. To do this, at each state Switcher first makes a *binary decision* to decide whether to activate its DEEPTHINK. Switching controls enable Switcher to learn at which states it ought to activate the DEEPTHINK model. Therefore, in LONDI, the Switcher agent uses a form of policies known as *switching controls* (Mguni et al., 2023b; Mguni, 2018). This leads to an RL problem in which, unlike the standard setup of an MDP, the Switcher agent now uses *switching controls* to select its decisions.

Summary of events



((a)) Task Map ((b)) budget=2 ((c)) budget=4 ((d)) budget=6 ((e)) budget=8

Figure 1. (a). The ScienceWorld task, Create a circuit. To complete the task, the agent must navigate to the hallway first and then determine the correct room to enter then use the material present in the room. (b),(c),(d),(e) sub-figures are the heatmap of LONDI DEEPTHINK calls with different budget on ScienceWorld task, Create a circuit. Since the agent needs to pass through the hallway to reach the workshop initially, LONDI must activate DEEPTHINK model in the hallway at least once. Subsequently, the remaining budget allocated to activating the DEEPTHINK model at the workshop to accomplish the task and obtain a higher reward. Therefore, as the budget decrease, LONDI focuses its activations of DEEPTHINK solely at the critical parts of the map such as hallway, resulting in lighter colors in the heatmap.

At a time $t \in 0, 1...$
• Encoder process the state $s_t \in \mathcal{S}$
• Switcher decides whether to activate the DEEPTHINK model according to the decision $g \sim \mathfrak{g} : \mathcal{S} \to \{0, 1\}$:
• if $g = 0$:
◦ The QUICK is activated and samples an action a_t from its policy π^{QUICK}. The switch receives a reward $r \sim \mathcal{R}(s_t, a_t)$ and the system shift to the subsequent state s_{t+1}
• if $g = 1$:
◦ DEEPTHINK is activated and samples an action a_t from its policy π^{DEEP}. Switcher receives a reward $r + c$ where $r \sim \mathcal{R}(s_t, a_t)$ and the system transitions to the state s_{t+1}.

We now describe how at each state Switcher decides whether to activate DEEPTHINK. At any s_t, the decision to turn the DEEPTHINK model is decided by a (categorical) policy $\mathfrak{g} : \mathcal{S} \to \{0, 1\}$. We denote by $\{\tau_k\}$ the times that activation takes place, for example, if the DEEPTHINK model is first activated at state s_5 then turned off at s_7, then $\tau_1 = 5$ and $\tau_2 = 7$. Recalling the role of $\mathfrak{g}$, the switching times obey the expression $\tau_k = \inf\{t > \tau_{k-1} | s_t \in \mathcal{S}, \mathfrak{g}(s_t) = 1\}$ and are therefore ***rules*** *that depend on the state.*. The termination times $\{\tau_{2k-1}\}$ occur according to some external (probabilistic) rule i.e., if at state s_t DEEPTHINK is active, then DEEPTHINK deactivates at state s_{t+1} with probability $p \in]0, 1]$. Hence, by learning an optimal $\mathfrak{g}$, Switcher learns the best states to activate DEEPTHINK.

3. Experiments

3.1. Experimental Setup

We conducted a series of experiments to test whether LONDI: **1.** Solves complex interactive tasks while reducing the computational cost. **2.** Works within the budget limit. **3.** Develops the ability to optimise its utilisation of DEEPTHINK within a specified budget for DEEPTHINK calls. **4.** Is plug & play, namely robust and consistent with different components. We use ScienceWorld (Wang et al., 2022) and BabyAI-Text (Carta et al., 2023) environments to test the performance of LONDI. We employ SAC (Haarnoja et al., 2018) to learn the control policy for switching. The presented plots display the average results obtained from 5 different seeds.

We evaluated the effectiveness of LONDI in two environments, namely Scienceworld and BabyAI-Text. Additional information regarding benchmark description and experimental setups is shown in G. We use a pre-trained Flan-T5 small model as our QUICK model and a pre-trained Flan-T5 large model (Chung et al., 2022) as our DEEPTHINK model. The baselines are SWIFTSAGE, FrugalGPT and Probabilistic policy. The details of the baselines are shown in Appendix G.5. Ablation studies are displayed in Appendix E to verify LONDI is robust to different components and the effectiveness of switching control structure.

3.2. Results and Analysis

Switching cost variation. To evaluate the performance of LONDI, we verified it with different cost values on different benchmarks. The results indicate that the cost c enables the resource/performance trade-off to be carefully calibrated by LONDI. With small cost, LONDI is willing to activate DEEPTHINK more and achieve higher reward. But when the cost is expensive, LONDI becomes extremely economical and rarely use DEEPTHINK model. Therefore, LONDI in this situation uses QUICK model mostly and the reward is reduced. The results shown in the Tables 1 and 2 suggest that as the computational cost increases, LONDI's performance declines, implying that higher costs lead to fewer invocations of the resource-intensive DEEPTHINK model.

Model	Reward
DeepTHINK only	77.6 ± 2.3
LONDI (cost=0.1)	76.5±3.4
LONDI (cost=0.2)	73.3±3.7
LONDI (cost=0.3)	49.6±2.6
LONDI (cost=0.4)	43.3±2.0
QUICK only	43.2±1.7

Table 1. Performance of LONDI on ScienceWorld task: *Identify Longest-then-shortest-lived animal* with different cost (normalized)

Model	Reward	Success Rate
DEEPTHINK only	0.96±0.02	0.87±0.05
LONDI (cost=0.05)	0.91±0.03	0.79±0.05
LONDI (cost=0.15)	0.82±0.05	0.72±0.07
LONDI(cost=0.25)	0.69±0.05	0.51 ±0.08
LONDI (cost=0.35)	0.51±0.03	0.36±0.04
QUICK only	0.49±0.01	0.34±0.02

Table 2. Performance of LONDI on BabyAI-Text with mixed tasks. The cost and reward are normalised values.

Budget version of LONDI. LONDI-B possesses a more straightforward adjustable parameter which directly constrains the utilization of the DEEPTHINK model, thereby limiting the computational expenses of the module. To assess the computational resource requirements, we measure the average performance of LONDI-B across five evaluation episodes and monitor the average GPU usage. The areas under the GPU usage curve are calculated based on numpy functions. The results are shown in Table 3. The DEEPTHINK calls in all LONDI-B structures remain within the specified budget. With a higher budget, the DEEPTHINK module is activated more frequently and the performance increases correspondingly. The computation metrics AUC indicate that LONDI-B significantly decreases the computational resource usage compared with only using the large language model. Specifically, LONDI with a budget of 2 achieves 90% performance compared with DEEPTHINK only but calls the DEEPTHINK model almost 60% fewer times.

Comparison to Baselines. To compare the performance of LONDI-B with baselines, we modify the QUICK and DEEPTHINK modules of SWIFTSAGE into the same setting, namely Flan-T5 small and large. The budget for calling DEEPTHINK model is set to be half of the calls of DEEPTHINK only in all tasks for LONDI-B and baselines. The results shown in Table 4 indicate that LONDI-B's performance is marginally inferior to DEEPTHINK only, but outperforms SWIFTSAGE, FrugalGPT and probabilistic policy in both baselines. In addition, considering that LONDI-B only calls the DEEPTHINK model half than DEEPTHINK only, it significantly minimize the computational cost of the system. More detailed information can be found in Appendix 7. [1]

Table 3. Computational cost of LONDI-B on the ScienceWorld task: *Identify Longest-then-shortest-lived animal*. DEEPTHINK Calls column represents the relative percentage of DEEPTHINK activations compared with DEEPTHINK only. AUC column gives the area under the GPU usage curve for the same number of timesteps.

Model	DEEPTHINK Calls	Reward	AUC
DEEPTHINK only	5±0.00(1)	77.6±2.3	3130±5
LONDI-B (budget=5)	4.87±0.04 (0.97)	76.5±3.3	2968±45
LONDI-B (budget=4)	3.96±0.03 (0.79)	75.3±3.5	2843±53
LONDI-B (budget=3)	2.82±0.04 (0.56)	72.3±3.7	2759±47
LONDI-B (budget=2)	1.72±0.05 (0.34)	70.6±3.2	2671±43
LONDI-B (budget=1)	0.83±0.02 (0.17)	65.7±2.1	2593±32
QUICK only	0±0.00	43.3±1.7	2451±2

Table 4. Average performance of LONDI-B and baselines on ScienceWorld and BabyAI mixed tasks. The reward value is normalized.

Model	ScienceWorld	BabyAI
DEEPTHINK only	0.87±0.02	0.96±0.02
LONDI-B	0.72±0.06	0.87±0.05
SWIFTSAGE	0.68±0.04	0.43±0.02
FrugalGPT	0.64±0.03	0.75±0.03
Probabilistic policy	0.44±0.08	0.41±0.07
QUICK only	0.33±0.01	0.34±0.01

4. Conclusion

In this paper, we introduce LONDI, a novel framework that leverages performance and computational cost by selectively activating LLM to cooperate with LM. LONDI combines LM and LLM in a way that enables LLM to support LM, thereby improving its performance. Simultaneously, LONDI can assist LLM in reducing computational and energy consumption. The budget variant of LONDI, known as LONDI-B, enhances the combination by offering an intuitive control facility for the user to limit the number of LLM calls. In our empirical investigations, we conducted a comprehensive set of experiments in ScienceWorld and BabyAI-Text. Across these domains, LONDI shows performance improvements and computational cost decreases.

[1]The switching policy employed by SWIFTSAGE is based on predefined rules and empirical observations specific to the ScienceWorld environment. It becomes challenging for this approach to perform effectively when applied to other environments or domains that may have different dynamics and characteristics.

Impact Statement

The objective of this paper is to reduce the overall computational consumption of LLM in addressing complex interactive tasks. One possible application of our work is to assist the downstream of LLM to edge devices, whose computational resources and energy consumption are limited. With LONDI, the edge devices can apply an affordable small LLM locally as QUICK and consider the cloud server as DEEPTHINK. Therefore, the edge devices can communicate to the cloud server for support only at necessary states, which minimises the overall consumption of energy and computational resources. This approach enables edge devices to achieve improved real-time decision-making capabilities, ensuring bandwidth and energy efficiency. Additionally, it could enables individuals to have an enhanced AI experience that is accessible to all with edge devices, ensures privacy through local data processing, and allows for customization and personalisation.

References

Bernoulli Distribution, pp. 36–37. Springer New York, New York, NY, 2008. ISBN 978-0-387-32833-1. doi: 10.1007/978-0-387-32833-1_24. URL https://doi.org/10.1007/978-0-387-32833-1_24.

Ahmed, N. and Wahed, M. The de-democratization of ai: Deep learning and the compute divide in artificial intelligence research, 2020.

Ahn, M., Brohan, A., Brown, N., Chebotar, Y., Cortes, O., David, B., Finn, C., Fu, C., Gopalakrishnan, K., Hausman, K., Herzog, A., Ho, D., Hsu, J., Ibarz, J., Ichter, B., Irpan, A., Jang, E., Ruano, R. J., Jeffrey, K., Jesmonth, S., Joshi, N. J., Julian, R., Kalashnikov, D., Kuang, Y., Lee, K.-H., Levine, S., Lu, Y., Luu, L., Parada, C., Pastor, P., Quiambao, J., Rao, K., Rettinghouse, J., Reyes, D., Sermanet, P., Sievers, N., Tan, C., Toshev, A., Vanhoucke, V., Xia, F., Xiao, T., Xu, P., Xu, S., Yan, M., and Zeng, A. Do as i can, not as i say: Grounding language in robotic affordances, 2022.

Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33: 1877–1901, 2020.

Caines, A., Benedetto, L., Taslimipoor, S., Davis, C., Gao, Y., Andersen, O., Yuan, Z., Elliott, M., Moore, R., Bryant, C., Rei, M., Yannakoudakis, H., Mullooly, A., Nicholls, D., and Buttery, P. On the application of large language models for language teaching and assessment technology, 2023.

Carta, T., Romac, C., Wolf, T., Lamprier, S., Sigaud, O., and Oudeyer, P.-Y. Grounding large language models in interactive environments with online reinforcement learning, 2023.

Central, H. Beating google and apple, huawei brings large ai model to mobile voice assistant, 2023. URL https://www.huaweicentral.com/beating-google-and-apple-huawei-brings-large-ai-model-to-mobile-voice-assistant/.

Chen, L., Zaharia, M., and Zou, J. Frugalgpt: How to use large language models while reducing cost and improving performance, 2023.

Chevalier-Boisvert, M., Bahdanau, D., Lahlou, S., Willems, L., Saharia, C., Nguyen, T. H., and Bengio, Y. Babyai: A platform to study the sample efficiency of grounded language learning, 2019.

Chung, H. W., Hou, L., Longpre, S., Zoph, B., Tay, Y., Fedus, W., Li, Y., Wang, X., Dehghani, M., Brahma, S., Webson, A., Gu, S. S., Dai, Z., Suzgun, M., Chen, X., Chowdhery, A., Castro-Ros, A., Pellat, M., Robinson, K., Valter, D., Narang, S., Mishra, G., Yu, A., Zhao, V., Huang, Y., Dai, A., Yu, H., Petrov, S., Chi, E. H., Dean, J., Devlin, J., Roberts, A., Zhou, D., Le, Q. V., and Wei, J. Scaling instruction-finetuned language models, 2022.

Ding, T., Chen, T., Zhu, H., Jiang, J., Zhong, Y., Zhou, J., Wang, G., Zhu, Z., Zharkov, I., and Liang, L. The efficiency spectrum of large language models: An algorithmic survey, 2023.

Feng, X., Luo, Y., Wang, Z., Tang, H., Yang, M., Shao, K., Mguni, D., Du, Y., and Wang, J. Chessgpt: Bridging policy learning and language modeling. *Advances in Neural Information Processing Systems*, 36, 2024.

Fu, D., Li, X., Wen, L., Dou, M., Cai, P., Shi, B., and Qiao, Y. Drive like a human: Rethinking autonomous driving with large language models, 2023.

Guan, L., Valmeekam, K., Sreedharan, S., and Kambhampati, S. Leveraging pre-trained large language models to construct and utilize world models for model-based task planning, 2023.

Haarnoja, T., Zhou, A., Abbeel, P., and Levine, S. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International conference on machine learning*, pp. 1861–1870. PMLR, 2018.

Huang, W., Abbeel, P., Pathak, D., and Mordatch, I. Language models as zero-shot planners: Extracting actionable knowledge for embodied agents. In Chaudhuri, K., Jegelka, S., Song, L., Szepesvari, C., Niu, G., and Sabato,

S. (eds.), *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pp. 9118–9147. PMLR, 17–23 Jul 2022.

Kahneman, D. *Thinking, Fast and Slow*. Farrar, Straus and Giroux, New York, 2011. ISBN 978-0-374-27563-1.

Lin, B. Y., Fu, Y., Yang, K., Brahman, F., Huang, S., Bhagavatula, C., Ammanabrolu, P., Choi, Y., and Ren, X. Swiftsage: A generative agent with fast and slow thinking for complex interactive tasks, 2023a.

Lin, B. Y., Huang, C., Liu, Q., Gu, W., Sommerer, S., and Ren, X. On grounded planning for embodied tasks with language models. *Proceedings of the AAAI Conference on Artificial Intelligence*, 37(11):13192–13200, Jun. 2023b. doi: 10.1609/aaai.v37i11.26549.

Lin, Z., Qu, G., Chen, Q., Chen, X., Chen, Z., and Huang, K. Pushing large language models to the 6g edge: Vision, challenges, and opportunities, 2023c.

Mguni, D. A viscosity approach to stochastic differential games of control and stopping involving impulsive control. *arXiv preprint arXiv:1803.11432*, 2018.

Mguni, D., Jafferjee, T., Wang, J., Slumbers, O., Perez-Nieves, N., Tong, F., Yang, L., Zhu, J., and Yang, Y. Ligs: Learnable intrinsic-reward generation selection for multi-agent learning. In *ICLR 2022-10th International Conference on Learning Representations*, volume 2022. ICLR, 2022.

Mguni, D., Jafferjee, T., Wang, J., Perez-Nieves, N., Song, W., Tong, F., Taylor, M., Yang, T., Dai, Z., Chen, H., et al. Learning to shape rewards using a game of two partners. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pp. 11604–11612, 2023a.

Mguni, D., Sootla, A., Ziomek, J., Slumbers, O., Dai, Z., Shao, K., and Wang, J. Timing is Everything: Learning to act selectively with costly actions and budgetary constraints. In *In International Conference on Learning Representations*, 2023b.

Mguni, D. H., Chen, H., Jafferjee, T., Wang, J., Yue, L., Feng, X., Mcaleer, S. M., Tong, F., Wang, J., and Yang, Y. MANSA: Learning fast and slow in multi-agent systems. In Krause, A., Brunskill, E., Cho, K., Engelhardt, B., Sabato, S., and Scarlett, J. (eds.), *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 24631–24658. PMLR, 23–29 Jul 2023c.

Puterman, M. L. *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons, 2014.

Qualcomm. World's 1st on-device stable diffusion on android, 2023. URL https://www.qualcomm.com/news/onq/2023/02/worlds-first-on-device-demonstration-of-stable-diffusion-on-android.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Wang, R., Jansen, P., Côté, M.-A., and Ammanabrolu, P. Scienceworld: Is your agent smarter than a 5th grader?, 2022.

Wang, Z., Cai, S., Chen, G., Liu, A., Ma, X., and Liang, Y. Describe, explain, plan and select: Interactive planning with large language models enables open-world multitask agents, 2023.

Wason, P. C. and Evans, J. S. Dual processes in reasoning? *Cognition*, 3:141–154, 1975.

Yi, R., Guo, L., Wei, S., Zhou, A., Wang, S., and Xu, M. Edgemoe: Fast on-device inference of moe-based large language models, 2023.

Appendix

A. Preliminary: Markov Decision Process

In this paper, we consider a setting in which an agent is tasked with solving a decision-making problem. We give some preliminaries on Markov decision processes which is the underlying formalism for our problem. A Markov decision process (MDP) (Puterman, 2014) is given by a tuple $\langle \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma \rangle$ where $\mathcal{S}$ represents the set of states, $\mathcal{A}$ means the set of (discrete) actions, the transition probability $\mathcal{P} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$ indicates the system dynamics, the reward function $\mathcal{R} : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ describes the performance of the agent, the reward discount factor $\gamma \in [0, 1]$ defines the level of discount applied to the rewards. At time t, the system is at state $s_t \in \mathcal{S}$ and the agent decides on its action $a_t \in \mathcal{A}$ using the policy $\pi : \mathcal{S} \times \mathcal{A} \to [0, 1]$, where $\pi(a|s)$ is the probability of choosing action $a \in \mathcal{A}$ under state $s \in \mathcal{S}$. The action transitions the system to a new state $s_{t+1} \sim \mathcal{P}(\cdot|s_t, a_t)$ and the agent then receives a scalar reward $r \sim \mathcal{R}(s_t, a_t)$. In the standard setup of an MDP, the agent's objective is to maximise the cumulative expected rewards $v^\pi(s) := \mathbb{E}[\sum_{t=0}^{+\infty} \gamma^t R(s_t, a_t)|a_t \sim \pi(\cdot|s_t)]$ using a policy $\pi^* \in \Pi$ where Π is the policy set of the agent.

B. Related Work

Recently, various tools have been proposed to augment the capabilities of LLMs. SayCan (Ahn et al., 2022) employs an LLM with an additional value function to assign scores to high-level actions, and then utilises a low-level planner to map these actions to determine their feasibility in the physical world. (Lin et al., 2023b) proposes an encoder-decoder structure to facilitate the planning ability of LM. (Huang et al., 2022) decomposes tasks into mid-level plans and maps outputs to available actions. DEST (Wang et al., 2023) applies a self-explanation mechanism for error correction and a goal-selector to rank sub-goals. Combining with PDDL, (Guan et al., 2023) utilises LLM to generate, translate and validate PDDL models to address planning tasks. Combined with reinforcement learning, GFlan (Carta et al., 2023) employs Flan-T5 (Chung et al., 2022) as the action policy and updates it with online PPO algorithm (Schulman et al., 2017). However, all these methods encounter the problem of high computational resource cost. To address the challenge of high computational costs, in this paper, we introduce a switching mechanism within a dual LM structure delineated by a low-cost and high-cost LM.

Closest to our work is the SWIFTSAGE (Lin et al., 2023a) framework, which combines a small LM module as the fast system and a large LLM module as the slow system. By combining two LLMs with varying sizes and computing power, the framework tackles intricate interactive reasoning tasks while mitigating the computational load. Although it achieves remarkable performance with GPT-4, the method to interpolate between the two modules uses a hand-crafted heuristic protocol which can lead to suboptimal performance (see Sec. 3.2). Another similar work is the FrugalGPT (Chen et al., 2023), which combines a cascade of LLMs and a score function to decide which LLM to use. However, defining an appropriate score function to guide the LLM selection process is challenging in complex planning tasks. Therefore, the computational constraint problem is imperfectly resolved in both cases. In comparison, LONDI uses reinforcement learning in conjunction with a type of policy known as *switching controls* to learn at which system states the DEEPTHINK module should be activated. Moreover, this systematic learning approach to the LLM activation enables a variant of LONDI to maintain a budget constraint on the number of DEEPTHINK calls.

The switching structure is similar to the mechanism of a psychological framework, dual process theory (Wason & Evans, 1975; Kahneman, 2011). Dual process structures have inspired various mechanisms in reinforcement learning used to improve learning efficiency. ROSA (Mguni et al., 2023a) and LIGS (Mguni et al., 2022) incorporate a dual switching method to activate a reward-shaping module to promote state visitations and coordination between adaptive agents in an RL and MARL respectively. LICRA (Mguni et al., 2023b) adds a trainable switch to decide whether to use a costly execution-policy system to generate actions. Similarly, MANSA (Mguni et al., 2023c) has an additional switch to decide whether to activate centralised training, a computationally expensive learning mode that facilitates coordination among adaptive agents.

C. Algorithm

We now describe the methodology for the LONDI framework. At a state s_t, the system initially checks the switch state m_t. If the switch is currently off, $m_t = 0$, the Switcher agent is applied to determine whether DEEPTHINK is needed. Otherwise

$m_t = 1$, the structure considers the switching probability p_i[2]. If $p_i(\cdot|s_t) = 1$, the switch keeps the same and the structure directly utilises DEEPTHINK to generate action. If $p_i(\cdot|s_t) = 0$, the structure activates Switcher agent to decide whether to turn on the switch again. If the Switcher module is used, it considers the state information s_t processed by the encoder and samples the action according to its discrete policy $g_t \sim \mathfrak{g} : \mathcal{S} \to 0, 1$. If the Switcher output 0, $g_t = 0$, the switch is off, $m_t = 0$, and the QUICK model is called to generate action and interact with the environment. Otherwise $g_t = 1$, the switch is on, $m_t = 1$, and the DEEPTHINK model is activated to generate action. The trajectories of the process are stored in a replay buffer for further training. To further explain the work flow of LONDI, a schematic is shown in Figure 2. In the budget-constrained variant LONDI-B, the key distinction lies in the replacement of the cost function with a parameter budget n, effectively limiting the computational resources allotted for the task. The switching control policy employed by LONDI-B takes into account the remaining computational budget, adjusting its decision-making process accordingly by: $g_t \sim \mathfrak{g}(\cdot|s_t, n)$. If the switching control mechanism invokes operations that surpass the predetermined budget limitations, the reward in and after that step will be reduced by a budget penalty n_p , discouraging such over-expenditure of resources. Instead of completely prohibiting the agent from accessing the DEEPTHINK model after exhausting budget, we apply this additional penalty n_p approach because it allows the agent to explore and consider information from later stages. If a particular point turns out to be highly important despite incurring a negative reward initially, the agent can strategically allocate a portion of its budget to access that point during subsequent training iterations. By applying a penalty rather than an outright restriction, the agent retains the flexibility to reevaluate and potentially utilize information from points that may become more valuable in later stages of the process.

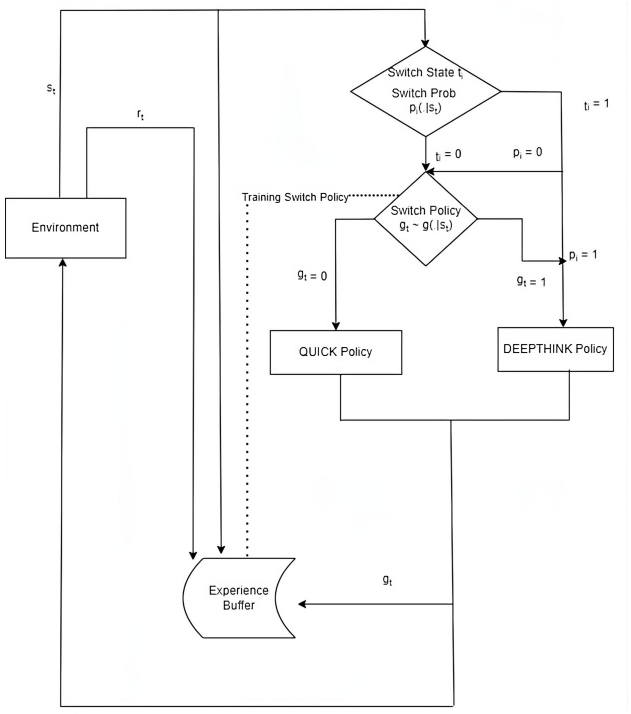


Figure 2. The schematic of LONDI. The Diamond represents the decision point, the square means a process or action, and the oval-like shape means data storage. The Switcher agent receives an environmental observation and makes a decision on which LLM module to utilise based on factors such as switch state, switch probability, and observation. The transition is then stored in the buffer for training the Switcher policy in subsequent iterations.

[2]The probability here is a Bernoulli distribution $p(x) = p^x(1-p)^{1-x}$ and the parameter p and $q = 1 - p$ of the distribution is adjustable based on the difficulty of the environment. Considering that the DEEPTHINL model in our setting (Flan model) cannot undertake long step planning, the switching probability p_i is applied ensures the possibility of sustained activation of the DEEPTHINK model, which improves the performance in complex and sparse reward tasks. $p_i(\cdot|s_t)$ means the value of p_i at state s_t, whose value is sampled from the Bernoulli distribution.

Algorithm 1 **L**anguage **O**ptimising **N**etwork **Di**stribution (LONDI)

Input: QUICK policy π^{QUICK}, DEEPTHINK policy π^{DEEP}, Switching Control Policy g, learning algorithm Δ^g, experience buffer **B**, switching probability p_i, switch state m_t, switch cost c
Output: Optimised policy g

```
Initialise g, p_i, m_t
for t = 1, T do
  Initialise m_t = 0
  while not done do
    Given environment state s_t evaluate g_t ~ 𝔤(·|s_t)
    if m_t > 0 then
      if p_i(·|s_t) = 1 then
        Sample action a_t using DEEPTHINK policy π^DEEP
      else
        if g_t = 1 then
          Sample action a_t using DEEPTHINK policy π^DEEP
        else
          Sample action a_t using QUICK policy π^QUICK, m_t = 0
        end if
      end if
    else if g_t = 1 then
      Sample action a_t using DEEPTHINK policy π^DEEP, m_t+ = 1
    else
      Sample action a_t using QUICK policy π^QUICK
    end if
    Apply a_t to environment to obtain s_{t+1}, τ_{t+1} and r_{t+1}
    if g_t = 1 then
      r_{t+1}− = c
    end if
    Store (s_t,, g_t, r_{t+1}, s_{t+1}) in B
  end while
  for Epochs and Batch numbers do
    Sample B to obtain (s_t,g_t,r_{t+1},s_{t+1}) and update 𝔤 with Δ^g
  end for
end for
```

D. Architecture and of LONDI

We now describe a concrete realisation of LONDI's core components which consist of 2 language models, a large language model (LLM) DEEPTHINK, a small language model as QUICK and a switching control RL algorithm as Switcher. Each component (including the LLMs) can be replaced by various other components.

• **QUICK model.** In this paper, we use a pretrained Flan-T5-small model (Chung et al., 2022) as the QUICK module.

• **DEEPTHINK model**. We use a pre-trained Flan-T5-large model (Chung et al., 2022) as the DEEPTHINK module. It performs twice as well as the QUICK module on average.[3]

• **Switching Control Policy (Mguni et al., 2023b).** A soft actor-critic (SAC) (Haarnoja et al., 2018) agent called Switcher whose policy's action set consists of 2 actions: 1) call the DEEPTHINK LLM 2) call the QUICK model.

• **Switching Control Encoder.** To enable SAC to perform in a textual environment we introduce an encoder to process text information. It consists of a transformer to turn text into a matrix and an MLP to condense information.

[3]The performance comparison is under our experiment setting, not the general comparison in all environments.

Algorithm 2 **L**anguage **O**ptimising **N**etwork **Di**stribution-**B**udget (LONDI-B)

Input: QUICK policy π^{QUICK}, DEEPTHINK policy π^{DEEP}, Switching Control Policy g, learning algorithm Δ^g, experience buffer **B**, switching probability p_i, switch state m_t, switch budget n, switch budget limit n_0, switch budget penalty n_p, switch cost c
Output: Optimised policy g

```
Initialise g, p_i
for t = 1, T do
  m_t = 0, n = n_0,
  while not done do
    Given environment state s_t and switch budget n, evaluate g_t ~ 𝔤(·|s_t, n)
    if m_t > 0 then
      if p_i(·|s_t) = 1 then
        Sample action a_t using DEEPTHINK policy π^DEEP
      else
        if g_t = 1 then
          Sample action a_t using DEEPTHINK policy π^DEEP, n− = 1
        else
          Sample action a_t using QUICK policy π^QUICK, m_t = 0
        end if
      end if
    else if g_t = 1 then
      Sample action a_t using DEEPTHINK policy π^DEEP, m_t+ = 1, n− = 1
    else
      Sample action a_t using QUICK policy π^QUICK
    end if
    Apply a_t to environment to obtain s_{t+1}, τ_{t+1} and r_{t+1}
    if g_t = 1 then
      r_{t+1}− = c
    end if
    if n < 0 then
      r_{t+1}− = n_p
    end if
    Store (s_t,, g_t, r_{t+1}, s_{t+1},n) in B
  end while
  for Epochs and Batch numbers do
    Sample B to obtain (s_t,g_t,r_{t+1},s_{t+1},n) and update 𝔤 with Δ^g
  end for
end for
```

E. Ablation Study

We conducted a series of ablation studies to verify the effectiveness of LONDI. In the subsequent analysis, we made adjustments to various elements of our framework to substantiate the following assertions:

LONDI effectively adapts to different components. To validate the dynamic modification capability of LONDI in activating the DEEPTHINK module based on the QUICK module's capability, we replaced the QUICK module with both a random agent and a more proficient FLAN-T5-small model. As shown in Figure 3 and Table 5, the results indicate that LONDI learns to activate the DEEPTHINK module according to the QUICK's performance. With a random QUICK module and a lower-performance QUICK module, LONDI can still achieve similar performance by activating the DEEPTHINK module more frequently. With lower performance DEEPTHINK module Flan-T5-small, LONDI shows the same tendency with various costs, which indicates that LONDI is a plug-in structure that can be used with distinct LLMs by only adjusting a few hyperparameters.

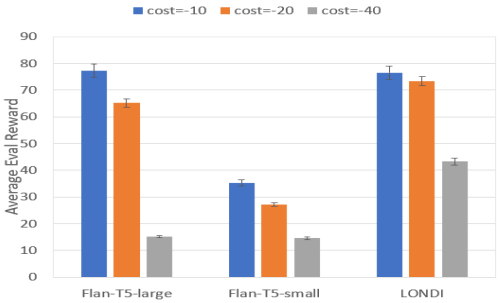


Figure 3. Performance of LONDI with a random agent as QUICK module on ScienceWorld task: *Identify Longest-then-shortest-lived animal*. The Flan-T5-large bar represents Flan-T5-large as the DEEPTHINK module. The Flan-T5-small bar represents Flan-T5-small as the QUICK module.

Table 5. Performance of LONDI with different QUICK modules on ScienceWorld task: *Create a circuit*. LONDI(L) means we modify the QUICK module to another FLAN-T5-small model which has a longer training length on this task.

Model	Reward	Rel. DEEPTHINK Calls
DEEPTHINK only	70±2.0	1±0.00
LONDI(L)	56±3.5	0.3±0.02
LONDI	52±3.1	0.5±0.04
QUICK only	9±0.0	0±0.00
FLAN-T5-small-L	30±2.3	0±0.00

Switching Controls are important. A central component of LONDI is its switching control mechanism which determines when to activate the DEEPTHINK model. In particular, the switching control mechanism allows the Switcher agent to learn to activate the DEEPTHINK model only at states where it is needed to drive higher performance. To evaluate the importance of the switching control component and the effect of budget constrain, we compared the performance of LONDI-B with a variant of LONDI-B in which the switching control mechanism is replaced with probabilistic policy. Observe that activating the DEEPTHINK model at all states degenerates the method to DEEPTHINK and similarly, never activating the DEEPTHINK degenerates the framework to QUICK. Table 6 shows the comparison of the performances of the variants under different budgets. We examined the performance of the variants of LONDI-B on the task called identifying the longest-lived animal. The results indicate that having the switching control component and hence, the ability to learn an optimal switching control in LONDI-B produces a significantly better performance compared to simply activating DEEPTHINK with Bernoulli distribution while LONDI-B uses fewer DEEPTHINK calls.

Table 6. The performance of LONDI-B and a variant of LONDI-B with a random Switcher (Rand. Switcher) agent with different budgets on ScienceWorld task: *Identify Longest-lived animal*. Data in blue and brown are related to LONDI-B and the random variant LONDI-B resp. The budget usage column represents the DEEPTHINK calls of LONDI under the different budget settings, where the budget usage of variant LONDI-B is always equal to the setting number. The average row represents the mean value of structures whose budget greater than zero. LONDI-B outperforms the variant with a random agent for all budgets larger than zero.

Budget	LONDI-B	Probabilistic policy	Budget Usage
No limit	83.2±2.1	69.3±7.2	3.93±0.04\4.8±0.42
budget=4	81.6±4.2	68.2±6.9	3.82±0.06\4±0.00
budget=3	77.6±5.3	66.4±7.5	2.76±0.07\3±0.00
budget=2	75.8±6.1	59.7±8.1	1.87±0.09\2±0.00
budget=1	66.4±3.2	50.8±6.4	0.82±0.04\1±0.00
budget=0	42.3±1.6	42.3±1.6	0.02±0.01\0±0.00
Average	**75.4±4.1**	**61.2±6.1**	**2.32±0.05\2.5±0.00**

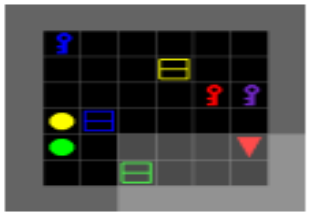


Figure 4. An illustration of one BabyAI task, PutNextLocal:"put the blue key next to the green ball". The shadow area represents the observable space of the agent.

F. Additional Results

The detail results of all tasks [4] on the Scienceworld are shown in Table 7. The cost of LONDI and QUICK is the proportion compared to the AUV of the DEEPTHINK model in the same timesteps. The results indicated that LONDI facilitates the collaboration between QUICK and DEEPTHINK models. The structure performs slightly worse than DEEPTHINK but significantly better than QUICK. However, LONDI requires only slightly more computational resources than QUICK only, while consuming significantly less than DEEPTHINK only.

Table 7. Detailed results on the ScienceWorld benchmark across different tasks. The budget or allowance for invoking the computationally intensive DEEPTHINK model is set to be half of the number of times the DEEPTHINK model would be called if it were the sole model employed.

Task name	LONDI-B	QUICK	DEEPTHINK	LONDI-B cost	QUICK cost	DEEPTHINK cost
Find an animal	78±4.1	23±0.4	100±0.0	0.74±0.02	0.68±0.01	1
Find a living thing	75±2.9	20±0.2	100±0.0	0.76±0.01	0.7±0.01	1
Find a non-living thing	78±3.2	58±1.7	100±0.0	0.72±0.02	0.69±0.00	1
Find plant	89±3.1	34±1.2	100±0.0	0.78±0.03	0.64±0.00	1
Inclined planes(determine angle)	52±3.2	10±0.0	73±2.1	0.80±0.02	0.74±0.01	1
Friction(known surfaces)	64±4.1	38±1.4	73±2.2	0.85±0.02	0.65±0.01	1
Identify Longest-then-shortest-lived animal	72±3.7	43±1.6	78±2.3	0.85±0.03	0.78±0.00	1
Identify Longest-lived animal	76±6.1	42±1.6	83±2.1	0.79±0.02	0.72±0.00	1
Identify shortest-lived animal	87±3.2	50±1.8	100±0.0	0.83±0.02	0.74±0.01	1
Create a circuit	52±3.1	9±0.0	70±2.5	0.79±0.02	0.62±0.01	1

G. Implementation Details

G.1. Benchmark descriptions

ScienceWorld. The ScienceWorld environment (Wang et al., 2022) simulates a residential setting comprising 10 interconnected areas with a diverse range of up to 200 objects, including devices, instruments, plants/animals, electrical components, substances, and containers, as well as common environmental items like furniture, books, and paintings. The action space in the ScienceWorld environment consists of 25 high-level actions, encompassing both science-specific actions and general actions. The agent can only observe the information of its current area. For different tasks, the agent needs to combine high-level actions and objects into applicable actions and receive periodic rewards if they move towards the goal. A sample illustration of one task: *Create a circuit* is displayed in Figure 1(a).

BabyAI-Text. BabyAI (Chevalier-Boisvert et al., 2019) is a 2D grid-based simulation environment that offers tasks of increasing complexity. The environment has various objects and the agent can pick up, drop, and move objects, and doors can be unlocked using keys of matching colors, which may be concealed within boxes. The agent's field of vision is limited to a 7x7 grid, and it cannot see beyond walls or closed doors. The available actions for the agent include moving forward, turning left or right, opening doors or boxes, picking up items, dropping items, and signaling completion. The agent can only hold one item at a time. The objective is to reach the goal state as quickly as possible, with the goal state being assigned a reward that diminishes over time. In this experiment, we use the modified textual version of BabyAI proposed by (Carta

[4] solvable by Flan-t5-large model

et al., 2023). One example task is shown in Figure 4.

G.2. Hyperparameter Settings

All hyperparameters used in our experiments are shown in the table below. The values included in square brackets indicate ranges of values that were used for performance tuning. All the training and evaluation is done on one NVIDIA A10 with 24GB memory. The training of LONDI with any version takes 8 hours under the setting of Table 8.

Table 8. Hyperparameter Setting of LONDI

Clip Gradient Norm	1
γ	0.99
Learning rate	1×10^{-4}
Number of minibatches	4
Rollout length	128
Number of optimisation epochs	4
Optimisation algorithm	Adam
τ	5×10^{-3}
ϵ	1×10^{-8}
Encoder MLP layer	1
Encoder MLP hidden unit	256
Use Generalised Advantage Estimation	True
Coefficient of switch cost	[-1,-5,-10,-15,-20,-25,-30,-40]
Switch budget penalty	[-25,-45,-65]
Encoder output size	[4,8,16]
Switching probability	[0.1,0,3,0.5,0.7,0.9]
Switch budget penalty	0.1(normalized)

G.3. Training Details of Flan models

Following the training setup of (Lin et al., 2023a) in ScienceWorld, we utilize flan-t5-large (783m) and flan-t5-small (77m) as the foundation, and fine-tuned them using the seq2seq action-prediction data (62k). In order to mitigate the potential bias arising from data imbalance in the sequence-to-sequence learning process, we employed a down-sampling technique, selectively reducing the representation of certain task types and actions, thereby curating a more balanced final dataset for the training phase. The training configs are consistent with (Lin et al., 2023a), a learning rate of 1e-4 and batch size of 128 employed for training 500 steps. The detailed information of the dataset is shown in Figure 5.

For experiments in BabyAI-Text, we apply a variant of Flan-T5, GFlan small and large (Carta et al., 2023), as our QUICK and DEEPTHINK model. For the training of language models, we follow the training framework of (Carta et al., 2023), which utilizes a Python library *Lamorel* to enable the dispatching of calls to the deployed LLMs from a single line of code within the RL loop, requesting the probability of actions for all environments. Therefore, RL can call and communicate with all LLMs in parallel. In addition, *Lamorel* help the update of LLMs using RL algorithm (i.e. PPO) loss. The hyperparameters of PPO are shown below.

G.4. Prompt of DEEPTHINK model

The prompt here is used by DEEPTHINK in ScienceWorld, the structure follows the setpup of (Lin et al., 2023a), which has planning stage and grounding stage. The planning stage includes a concise summary of the task description and the sequence of previous actions, followed by posing five critical questions related to the current state.

- "To complete the task, which objects do I need to collect? Please list them and their possible locations one by one."
- "Are there any objects that have not been collected yet?"
- "To complete the task most efficiently, what are the important subgoals to achieve? Please list the subgoals one by one."
- "Considering these subgoals, what have I already completed? And which subgoal should I focus on right now?"
- "Have I made any mistakes that might prevent me from efficiently completing the next subgoal? If any, how should

Task Type	Topic	Name	*Lens	#Vars: Train	Dev	Test	# Actions
1-1	Matter	Changes of State (Boiling)	107.7	14	7	9	694
1-2	Matter	Changes of State (Melting)	78.6	14	7	9	427
1-3	Matter	Changes of State (Freezing)	88.9	14	7	9	469
1-4	Matter	Changes of State (Any)	75.2	14	7	9	344
2-1	Measurement	Use Thermometer	21.4	270	10	10	4278
2-2	Measurement	Measuring Boiling Point (known)	35.2	218	10	10	6511
2-3	Measurement	Measuring Boiling Point (unknown)	65	150	10	10	9768
3-1	Electricity	Create a circuit	13.6	10	5	5	94
3-2	Electricity	Renewable vs Non-renewable Energy	20.8	10	5	5	169
3-3	Electricity	Test Conductivity (known)	25.6	48	10	10	1341
3-4	Electricity	Test Conductivity (unknown)	29	300	10	10	6974
4-1	Classification	Find a living thing	14.6	150	10	10	1606
4-2	Classification	Find a non-living thing	8.8	150	10	10	756
4-3	Classification	Find a plant	12.6	150	10	10	1458
4-4	Classification	Find an animal	14.6	150	10	10	1606
5-1	Biology	Grow a plant	69.5	62	10	10	3675
5-2	Biology	Grow a fruit	79.6	62	10	10	4283
6-1	Chemistry	Mixing (generic)	33.6	16	8	8	347
6-2	Chemistry	Mixing paints (secondary colours)	15.1	18	9	9	224
6-3	Chemistry	Mixing paints (tertiary colours)	23	18	9	9	350
7-1	Biology	Identify longest-lived animal	7	62	10	10	298
7-2	Biology	Identify shortest-lived animal	7	62	10	10	298
7-3	Biology	Identify longest-then-shortest-lived animal	8	62	10	10	360
8-1	Biology	Identify life stages (plant)	40	6	3	5	165
8-2	Biology	Identify life stages (animal)	16.3	4	2	4	31
9-1	Forces	Inclined Planes (determine angle)	97	24	10	10	2733
9-2	Forces	Friction (known surfaces)	84.9	26	10	9	3644
9-3	Forces	Friction (unknown surfaces)	123.1	23	10	10	3284
10-1	Biology	Mendelian Genetics (known plants)	130.1	26	10	10	3043
10-2	Biology	Mendelian Genetics (unknown plants)	132.1	24	10	10	2853
		Short ($0 <$ *Len ≤ 20)	11.76	81.80	8.6	8.80	673.10
		Medium ($20 <$ *Len ≤ 50)	28.58	110.75	8.13	8.38	2516.88
		Long (*Len > 50)	94.30	37.75	9.00	9.58	2934.75
		Overall (avg)	49.26	71.90	8.63	9	2069.43
		Overall (sum)	N/A	2,157	259	270	62,083

Figure 5. The statistics of ScienceWorld benchmark. **Len* represents the average length of the trajectories of oracle agents. Number of downsample variations are shown in each split. The rightmost column represents the quantity of data points utilized for the action-prediction seq2seq task during the training phase of Flan model.

Table 9. Hyperparameter Setting of PPO in GFlan

Variables	**Values**
Number of transitions collected between two updates	320 (8 environments ×40 steps in each environment)
Number of epochs per update	4
Batch size	32
Entropy loss coefficient	0.01
Value function loss coefficient	0.5
Discount factor	0.99
Learning rate	1×10^{-6}
λ factor of the Generalized Advantage Estimator	0.99
Optimisation algorithm	Adam
Clipping parameter ϵ	0.2
Maximum gradient norm	0.5

I fix them?" The grounding stage includes a comprehensive list of supported action types in a formal manner at first, complemented by remarks. The output of the planning stage are then given as advice. Recent action history of the past 10 time steps are also given. With these information, the LLM is required to focus on the next subgoal and generate a list of actions to achieve it.

G.5. Baselines

We introduced all baseline methods and their experimental settings when we used to compare them with the LONDI as follows.

SWIFTSAGE(Lin et al., 2023a) utilizes a rule-based method to switch the activation between DEEPTHINK and QUICK model: 1) There are five consecutive time steps with zero reward. 2) The QUICK's prediction for the next action can result in a critical decision, such as giving the final answer for the experiment result. 3) The QUICK's prediction for the next action is invalid in the current environment or the observation of the action suggests that an exception is encountered. In this paper, we keep the DEEPTHINK and QUICK models of SWIFTSAGE consistent with those used in LONDI.

FrugalGPT(Chen et al., 2023) utilizes a cascade of LLMs and a score function to minimize the cost. The cascade tries with the smallest LLM first and the score function grades the response and query pair. If the score is higher than the threshold, the agent accept the answer. Otherwise, the cascade tries the larger LLM and the score function grades again until the answer meets the requirement or the largest LLM is used. In this experiment, we use the environment feedback as the score and set the threshold based on different tasks. In addition to the reward of environment, we added another reward that is proportional to the distance between the current location of agent and goal position. This was done to ensure more accurate feedback in alignment with the agent's actions. In this paper, the LLM cascade of FrugalGPT includes QUICK and DEEPTHINK models which are identical to those employed in LONDI.

Probabilistic policy is a policy that based on Bernoulli distribution (Ber, 2008): $P(x) = p^x(1-p)^{1-x}$. For the experiments, we configured the probability p to be 0.5, implying that the values 0 and 1 had an equal chance of being selected. When the sample value of the distribution is 1, the policy decides to use the DEEPTHINK model. Otherwise, the policy use the QUICK model.

H. Limitations

Although LONDI has shown adaptability to other environments, the trainable model free switcher limits its applicability to more resources constraint environments since the model-free algorithm requires large sample data for training which is not applicable in some real-world situation. One possible solution is to use model-based algorithms instead, which could alleviate sample efficiency problem. In addition, LONDI cannot be directly generalized to another unseen environments without training, which could reduce its application scope in real life. One possible approach is to combine reinforcement learning with causal inference to achieve better generalizability.