Nash CoT: Multi-Path Inference with Preference Equilibrium

Anonymous ACL submission

Abstract

Chain-of-thought (CoT) prompting has emerged as a powerful technique for enhancing the reasoning capabilities of Large Language Models (LLMs) on complex problems. Among CoT-related studies, self-consistency (Multi-path inference with answer filtering through voting) involves generating multiple reasoning paths using the CoT framework and then selecting the most frequently produced outputs standing out as a concise yet competitive approach. While self-consistency has indeed led to the improvements in LLM inference, the use of multi-path inference also escalates deployment costs. Therefore, maintaining the performance benefits of self-consistency inherited from multi-path inference while reducing the inference costs holds significant value. In this research, we conceptualize language decoding as a preference consensus game, constructing a bi-player gaming system within each local path, and introduce Nash Chain-of-Thought (Nash CoT). Specifically, for a given question, we leverage LLM to autonomously select the contextually relevant template and generate outputs guided by this template, aiming to reach Nash Equilibrium alongside normal generation in each path. This approach allows us to achieve comparable or improved performance compared to self-consistency while using fewer inference paths on various inference tasks, including Arabic reasoning, Commonsense Question answering, and Symbolic inference.

1 Introduction

Large Language Models (LLMs) have revolutionized Natural Language Processing (NLP) (Ouyang et al., 2022; etc., 2023; Jiang et al., 2023; Brown et al., 2020b; OpenAI, 2024). In particular, leveraging human-designed instructions as input, LLM demonstrates superior inference performance across various types of simple reasoning tasks (Radford et al., 2019; Brown et al., 2020a). But, its performance remains variable in complex reasoning tasks (Rae et al., 2022). To enhance LLM's inference capabilities on complex issues, we can employ a step-by-step inference approach, known as Chain-of-Thought (CoT) prompting (Wei et al., 2023). For instance, starting with a template like `"Let's think step by step"`, LLM first generates rationales and then arrives at a final prediction. This approach significantly improves LLM's inference performance across tasks like Arabic, Symbol Inference, and CommonsenseQA.

Subsequently, the impressive performance of CoT on complex inference tasks has spurred new developments in this direction (Wang et al., 2023; Wei et al., 2023; Jin and Lu, 2023; Zhang et al., 2022; Shi et al., 2022). Among these developments, self-consistency (Wang et al., 2023) emerges as a competitive CoT approach. It leverages uncertainty from **multiple inference paths** from a single LLM, and ranking generated answers by frequency can significantly enhance LLM's inference performance. This approach significantly improves the performance of GPT-3 utilizing zero-shot CoT without any options for parameter tuning. Meanwhile, experimental results indicate that inference performance improves with the number of generated samples (Wang et al., 2023), suggesting that the potential of LLM's inference capabilities has not yet to be fully exploited. Although self-consistency is straightforward to implement and requires no additional turning, it has a significant drawback: *substantially higher inference costs compared to directly utilizing CoT.*

On the other hand, the performance improvements resulting from self-consistency imply that relying solely on single-path inference cannot fully harness the inference capabilities of LLM, and multi-path inference encompasses the potential correct answers. Therefore, it's necessary to maintain this strategy. Meanwhile, numerous practical applications indicate that when a LLM is given

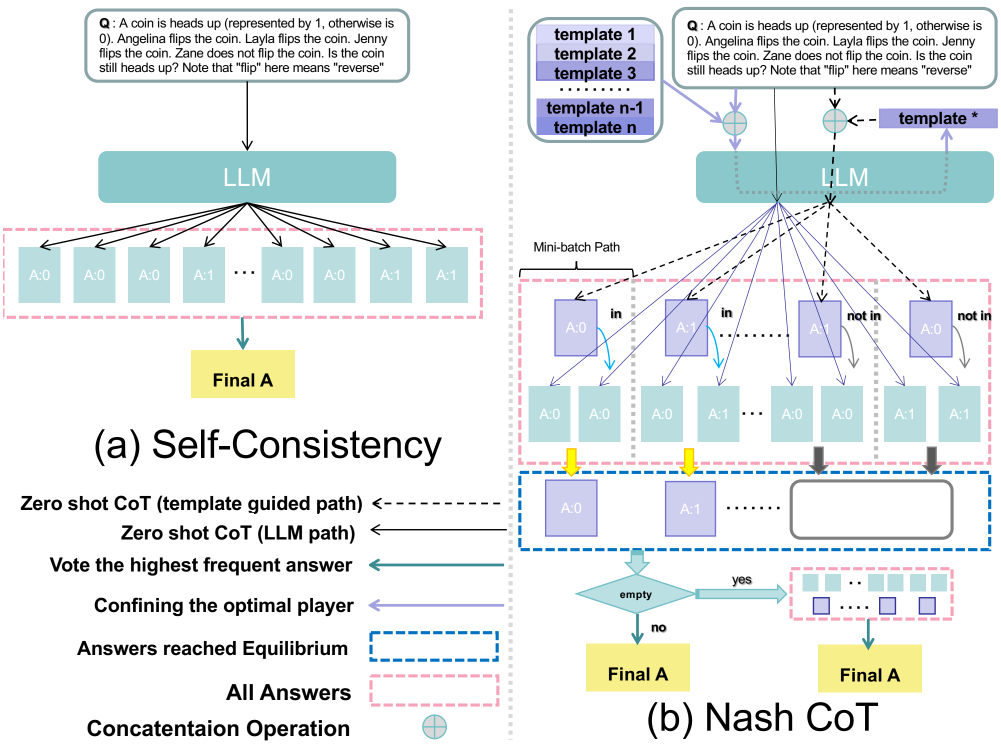

Figure 1: Demonstrations of Self-Consistency and Nash Chain-of-Thought (Nash CoT). (a) Self-Consistency entails inferring multiple paths and subsequently voting the most frequently prediction. (b) In Nash CoT, inference occurs multiple times, with each path generating two responses, but only one reached Preference Equilibrium is sustained. Ultimately, the answer sampled is the one that reaches preference equilibrium.

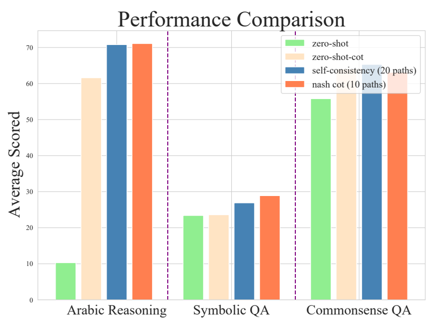

Figure 2: General Performance Comparison. We compare the average performance of , zero-shot, and zero-shot CoT self-consistency (20 Paths) with our Nash CoT (10 Paths) on Mistral-Instruct and GLM4.

the appropriate templates, it can perform specific tasks more proficiently. Hence, an intuitive strategy to reduce the number of inference paths in self-consistency is to use templates to guide the LLM to correctly infer each path.

To achieve this goal, we use the contextual information required by the question as a template to guide the inference in each path. This approach increases the probability that the LLM can correctly solve the question in each path, thereby potentially reducing the number of inference paths needed for multi-path inference. Meanwhile, to alleviate overconfidence in template-guided generation, we develop a bi-player gaming system that introduces the Nash Equilibrium of preference of strategy (defined as Preference Equilibrium) in multi-path inference. This system samples generations that balance the preferences of both template-guided and normal generation, ensuring the output robustly matches the context of the given question while sustaining some moderate generation pattern. Subsequently, we combine multi-path inference with Preference Equilibrium to propose **Nash CoT**. We present the comparison between Self-Consistency and Nash CoT in Figure 1.

We conduct experiments with two local deployed LLMs-Mistral-Instruct (Zeng et al., 2022; Du et al., 2022) and GLM4 (Zeng et al., 2022; Du et al., 2022) on various inference tasks, including Arabic Reasoning (Koncel-Kedziorski et al., 2015; Hosseini et al., 2014) and Symbolic Reasoning (Wei et al., 2023) and Commonsense Reasoning (Talmor et al., 2019; Geva et al., 2021). As shown in Figure 2, Nash CoT has achieved similar or even better performance against self-consistency with fewer inference paths. Meanwhile, as shown in Figure 3, Nash CoT has significantly reduced the inference cost by up to 50% on local deployed LLMs.

To our knowledge, we are the first to introduce Preference Equilibrium into multi-path inference with the aim of harmonizing text generation guided by the optimal template with that generated by the model's default state. It can reduce the number of paths needed to achieve good results in multi-path inference. And we are also the first to integrate Preference Equilibrium into multi-path inference and propose Nash CoT. It can achieve similar or even superior performance on various reasoning tasks while requiring only half costs compared to self-consistency.[1]

2 Related Work

Chain-of-Thought (CoT). There are three majority kinds of CoT approaches. Zero-shot CoT that Prompts LLM with simple yet instructions to guide LLM generate step by step (Kojima et al., 2023). Manual CoT that initialized by randomly sampled several cases from dataset or designed by human, and followed by utilizing these cases as demonstration to guided LLM generate follow the manner of demonstration (Wei et al., 2023), however, such methods can be biased if the demonstration can represent real distribution. Automatic (or batch) CoT (Zhang et al., 2022) first sample cases which have the most representative sentence embedding in each clusters, followed by inference with the same manner as manual CoT. Self-Consistency (Wang et al., 2023) showcases strong performance in vast benchmarks. Apart from its impact on inference performance,self-consistency also boasts scalability as a key advantage. It seamlessly integrates with different approaches, such as tabular consistency of transformations (tab-CoT) (Jin and Lu, 2023), making it adaptable and versatile for various applications. Despite self-consistency has improved LLM's performance on Arabic benchmarks. Correspondingly,self-consistency should have to inference multi-times, thus it burdens the deploy budgets.

One way to address this limitation is by initially sampling multiple inference paths and then fine-tuning using the highest frequency path. Specifically, Huang et al. (2022) suggest that gathering inferences from multiple paths and sampling the most frequent generation to fine-tune a smaller LLM can enhance the LLM's inference performance. However, this approach still requires updating the LLM's parameters, which is inefficient. Therefore, it is necessary to further explore inference methods to maintain the performance of self-consistency while reducing the number of inference paths.

[1]We will release our code at <Removed for Submission>.

Preference Optimization. The training policy with Reinforcement Learning (RL) to reflect preference, termed Reinforcement Learning with Human Feedback (RLFH), was initially introduced by (Akrour et al., 2011) and has since undergone consistent improvement and refinement by (Cheng et al., 2011; Busa-Fekete et al., 2013; Wilson et al., 2012). This approach has been widely applied to adjust Large Language Models' (LLMs) parameters to align with human preferences (Ouyang et al., 2022; Jiang et al., 2023; etc., 2023). Recently, a new approach called Direct Optimizing from Preference (DPO) has been proposed by (Rafailov et al., 2023), aiming to directly adjust LLMs to reflect human preferences without requiring a reward model and RL algorithm. Additionally, (Munos et al., 2023) proposed combining DPO with Nash equilibrium to ensure convergence of the last iterated policy. Our study also utilizes the concept of equilibrium in preference model, but the main difference is that we utilize preference equilibrium as a standard to pick up the most preferred answer instead optimizing the LLM's parameters.

3 Preference Equilibrium in mini-batch inference

Self-consistency has shown that the inference of LLMs under a single path may not represent their full capabilities. By simply conducting multiple inferences and filtering answers through voting, it is possible to achieve more accurate results. However, multi-path inference lacks a strong theoretical foundation to determine the optimal number of inference paths, potentially leading to much more computational resource consumption. To reduce the number of paths required for multi-path inference, we utilize the concept of Nash Equilibrium to locally construct a binary game system in multi-path inference. Specifically, the preference of each valid inference path of the LLM needs to achieve Nash equilibrium with the preferences of the generation guided by the template. This approach increases the probability of each path correctly answering the question while maintaining a certain level of robustness, thereby reducing the number of inference paths required by self-consistency.

Preference Model. Given text input x, and the sampled predictions (answers) y_1, y_2, we first define y_1 prefers y_2 as Equation 1:

$$\mathcal{P}(y_1 \prec y_2|x) := \text{sign}(r_\theta(y_1|x)) - \text{sign}(r_\theta(y_2|x)), \quad (1)$$

where r_θ denotes preference model that reflecting the preference when comes to the pairs. Then we imply the existence of Nash equilibrium in Preference model, specifically, Equation 1 is strictly linear, *i.e.* $\mathcal{P}(y_1 \prec y_2|x) = 1 - \mathcal{P}(y_2 \prec y_1|x)$.

Player Templates: Templates for our preference model that are shown the structure: `id(player): description`.

Mathematician: You are a mathematician, you excel at analyzing problems from a mathematical logical perspective and arrive at conclusions that align with your values.
Literary scholar: You are a literary scholar who has read a vast array of literary works. Please consider the problem from the perspective of a literary scholar.
Philosophical: You are a philosopher, your knowledge base includes a wealth of philosophical knowledge. You enjoy approaching problems from a philosophical perspective and arriving at conclusions that align with your values.
Geographer: You are a geographer with a deep understanding of geographical knowledge. Please approach the given problem from the perspective of a geographer.
`··· (other cases have been appended to the Appendix.)`

Additionally, drawing from the definition from Munos et al., we define one policy as more preferred over another as:

$$\mathcal{P}(\pi_1 \prec \pi_2) := \mathbb{E}_{\substack{y_1 \sim \pi_1(\cdot|x) \\ y_2 \sim \pi_2(\cdot|x)}}[\mathcal{P}(y_1 \prec y_2|x)] \quad (2)$$

Preference Equilibrium. We aim to select the best template (we provide several cases in above example 3 for the current problem, thus facilitating the large model in problem-solving, correspondingly reduces the required num of inference paths. However, this may lead to some issues: 1) If the template is incorrectly chosen, it may cause the agent to generate answers outside the range of correct responses corresponding to the current problem, resulting in erroneous replies. 2) The large model may excessively generate context-dependent responses, affecting its robustness. To address these issues, we build a local **bi-player gaming system** that the preference of template guided LLM (`player 1`) over normal status of LLM (`player 2`) is the pay-off of template guided LLM, vice visa. If `player 1` and `player 2` reaches Nash Equilibrium [2], then the strategy can match the preference of both `player 1` and `player 2`.

Subsequently, we define the status that `player 1` and `player 2` reach Nash Equilibrium as **Preference Equilibrium** (**Definition 1**). Meanwhile, in Theorem 3.1, we prove the existence of Nash Equilibrium in this system. Specifically, the strategy of `player 1` equal to `player 2` is one solution that this system has reached **Preference Equilibrium**.

Theorem 3.1 (Existence of Preference Equilibrium)**.** *Given any two policy (player) π_1 and π_2 within the gaming system defined in Definition 1, where $\pi \in \Pi$. $\pi_1 \equiv \pi_2$denotes a solution where the gaming system reaches Nash Equilibrium.*

Proof of Theorem 3.1 see Appendix E.

Meanings of the existence of Preference Equilibrium. Theorem 3.1 proves the existence of an Nash Equilibrium between the template guided LLM and the normal status of LLM. When reaching Preference Equilibrium, the preference of decisions made by the template guided LLM are aligned with those made by the LLM under normal status. Meanwhile, the preference of template guided LLM generation is much more closed to the requirement of quesitons' context, while those of the normal status LLM are predominantly based on its parameters which is much more robust than template guided LLM. Therefore, this equilibrium can balance the requirement of contextual information and robustness of the model generation during problem-solving. **Notabaly**, $\pi_1 \equiv \pi_2$ means their outputs are also likely to be equal. This insight forms a fundamental basis for piratically implementing Nash CoT.

3.1 Mini-batch inference with Preference Equilibrium

Subsequently, based on the concept of Preference Equilibrium, we conceptualize a Mini-batch infer-

[2]Preference Equilibrium leverages the concepts of Nash Equilibrium. Nash equilibrium is proposed by John Nash. It is a concept solution where, assuming each participant knows the equilibrium strategies of the other participants, no participant can benefit by changing their own strategy.

ence (shown in Figure 1) as a bi-player gaming system. This approach aims to achieve better inference compared to direct inference, while still preserving some of the inherent randomness of standard inference methods. To begin with proposing this system, we first define x^t as the template of zero-shot CoT, $\{x_0^c, x_1^c, \cdots, x_n^c\}$ (we provided several cases in Player Templates) as the candidates template for template guided generation. Meanwhile, in this system, the template can to be chosen by a reward model r_θ.

In terms of the process of mini-batch inference, we firstly inference LLM twice times (we have conducted ablations about 'twice' in section ablation) *i.e.* $[y_0, y_1] \leftarrow [\pi(\cdot|x^t, x), \pi(\cdot|x^t, x)]$. Meanwhile, due to the inherent uncertainty of LLM, the generation of $[y_0, y_1]$ can be considered a potential set of distinct predictions. Subsequently, the template guided generation can be sampled by querying LLM with x^c and x^t *i.e.* $y^* \leftarrow \pi(\cdot|x^c, x^t, x)$. Furthermore, we can select an answer from y_1 and y_2 that is the same as y^*, thereby satisfying the Nash Equilibrium described in Theorem 3.1. Based on the mini-batch inference, we further introduce Nash CoT in the next chapter. (**Notably,** the patterns in this chapter may not always hold true. For instance, y^* may not always in $[y_1, y_2]$. We will address this issue in the following chapters.)

4 Nash Chain of Thought (Nash CoT)

Algorithm 1 Nash CoT (Answer Gathering)

Require: Candidate question q sampled from $\mathcal{Q} = \{q_0, q_2, \cdots, q_n\}$; Outer iterations $\mathrm{n_{outer}}$; Num of mini-batch inference $\mathrm{n_{mini}}$; Large language model π; CoT prompt x^t, candidiate player template $\{x_0^c, x_1^c, \cdots, x_n^c\}$, $\mathrm{Prompt}(\{x^c\})$ is used to point out the most preferred x^c.

Generation:

Initialize answer list $\mathrm{ans} = []$.
$x_c \leftarrow \pi(\cdot|\mathrm{Prompt}(\{x^c\}))$
for t in range($\mathrm{n_{outer}}$) **do**
  Initialize prefer pairs $\mathrm{pref} = []$.
  for t in range($\mathrm{n_{mini}}$) **do**
    $y \leftarrow \pi(\cdot|x^t, q)$;
    $\mathrm{ans.append}(y)$;
  end for
  $y^* \leftarrow \pi(\cdot|x^c, x^t, x)$
  $\tau.\mathrm{append}([y^*, \mathrm{ans}])$
end for
Return τ

Nash CoT can be seen as an extension of Mini-batch inference with Preference Equilibrium, implementing multiple Mini-batch inferences to enhance performance. This approach is influenced by experimental results from self-consistency, which suggest that increasing the number of paths can improve inference accuracy. Meanwhile, the reasoning process for each question is divided into two stages: Answer Gathering and Answer Filtering.

Answer Gathering. When generating candidate answers, the process predominantly involves two types of loops: **Mini-batch Loops** ($\mathrm{n_{mini}}$): In Chapter 3.1, we discussed the implementation of mini-batch inference with Preference Equilibrium. As shown in Algorithm 1, this process involves searching for template-guided generations within two rounds of generation ($[y_1, y_2]$). We refer to the times of these two predictions as the $\mathrm{n_{mini}}$. Moreover, to mitigate the impact brought from low-frequency predictions, we introduce iterating $\mathrm{n_{mini}}$ multiple times. This leads us to another type of loop: **Outer Loops** ($\mathrm{n_{outer}}$): This loop resembles the concept of multi-path in self-consistency. After completing loop $\mathrm{n_{outer}}$, we filter the generated answers and retain the answer that reaches equilibrium most frequently (shown in Algorithm 2), as the preferred answer.

Answer Filtering. In terms of answer filtering, as shown in Algorithm 2 we first count the most frequent prediction satisfy Preference equilibrium. Specifically, we count all y^* satisfy $y^* \in [y_1, y_2]$ and compute their frequency. Subsequently, we return the most frequent case. Otherwise, if is no cases satisfy $y^* \in [y_1, y_2]$, we adopt the strategy of self-consistency by selecting the most frequent prediction among all generated answers.

Algorithm 2 Nash CoT (Answer Filtering)

Require: Preference pair list τ

Filtering:

Initialize hash table: $\mathrm{hash} = \{\} : k \to v$.
Initialize new answer list $\mathrm{nans} = []$.
for $[y^*, \mathrm{ans}]_i$ in τ **do**
  if $y^* \in \mathrm{ans}$ **then**
    $\mathrm{hash}[y^*]$+=1
  end if
  $\mathrm{nans.extend}([y^*, \mathrm{ans}[0], \mathrm{ans}[1]])$
end for
if $\mathrm{hash} \equiv \{\}$ **then**
  return the most frequent y in nans
else
  return $y \leftarrow k = \arg\max_v \mathrm{hash}$
end if

Subsequently, we propose Nash CoT, which iterates through Algorithm 1 and Algorithm 2 to perform inference on all sampled questions, where τ represents the candidate answers from the Answer Gathering stage.

Core LLM	Methods	SingleEQ	AddSub	MultiArith	GSM8K	AQuA	SVAMP	Avg.
Mistral-Instruct (7B)	zero-shot	15.3 ± 0.8	12.0 ± 2.8	3.3 ± 1.3	2.7 ± 2.0	20.8 ± 1.5	7.7 ± 2.0	10.3± 6.5
	zero-shot CoT	76.0 ± 0.8	82.5 ± 2.0	75.4 ± 6.1	44.3 ± 4.0	27.9 ± 2.3	63.4 ± 6.9	61.6±19
	self-consistency (20 Paths)	82.5 ± 0.8	86.3 ± 5.1	86.3 ± 2.8	58.5 ± 2.8	34.4 ± 6.1	76.5 ± 2.8	70.8± 19
	Nash CoT (10 Paths)	81.4 ± 0.8	86.3 ± 6.0	86.3 ± 4.7	55.7 ± 5.8	39.9 ± 5.4	77.0 ± 3.5	71.1± 17
GLM4-chat (9B)	zero-shot	1.1 ± 1.5	1.1 ± 1.5	12.6 ± 3.9	12.0 ± 2.0	22.4 ± 4.1	4.4 ± 2.8	8.9±7.6
	zero-shot CoT	90.7 ± 1.5	90.7 ± 1.5	98.4 ± 1.3	80.9 ± 2.8	20.8 ± 3.1	86.9 ± 3.5	78.1±26.1
	self-consistency (20 Paths)	92.3 ± 2.0	90.2 ± 2.3	98.4 ± 2.3	89.3 ± 0.2	20.8 ± 3.1	91.5 ± 1.1	80.4±26.8
	Nash CoT (10 Paths)	91.3 ± 0.8	90.2 ± 2.7	96.7 ±3.3	80.3± 1.3	20.8±3.1	88.0 ±2.0	77.9±26.1

Table 1: Experimental results on arithmetic reasoning benchmarks. We test Zero-Shot CoT and Nash CoT with the core LLM includes Mistral-Instruct (7B) and GLM4-chat (9B) on mathematical benchmarks including AddSub, MultiArith, SingleEQ, SVAMP, GSM8K, and AQuA. Nash CoT performs the best.

Preference Templates: Templates we utilized to confine the prompt for preference model.

Q: Current issue is **{query}**, and the best player is who? Please give us the number of that player from the options below: **{description}**. There are total **N({key(player)})** players including **{key(player)}**. Please point out the most appropriate player for the following task: **candidate questions**

A: Let us think step by step. $\rightarrow$ z
`// (obtain the rational z)`

A: Let us think step by step. + z+ Therefore, the most appropriate player in this game is who? (please direct give us the number)
`// (obtain the answer)`

Practical Implementation of reward model r_θ. In the process of practical implementation, we do not explicitly train a reward model r_θ to confine the player template x^c (we have provided cases in **Player Templates**) using Equation 1. Instead, we directly use the preference template (shown in **Preference Template**) to guide the LLM in determining the most suitable player template for a given question. For example, when presented with a coin flip question as shown in Figure 4, we fill the **Preference Template** with given question and player templates. This filled template is then input into the LLM to provide the id of the most suitable player template from the available options. In particular, we believe it's effective, this is because most of baselines we selected have been turned to reflect human preference, thus we believe the selected LLM can be directly utilized as the preference model to point out the most preferred option among candidate options.

5 Experiments

The goal of our experiment is to 1) demonstrate the performance advantage and effectiveness of Nash CoT. 2) shocase whether Nash CoT help reduce the overall inference time. In the following sections, we first introduce our experimental setup and then present the experimental results and analysis.

Datasets. Our majority benchmarks are composed of three different kinds of inference tasks. 1) *arithmetic reasoning:* SingleEq (Koncel-Kedziorski et al., 2015), AddSub (Hosseini et al., 2014), MultiArith (Roy and Roth, 2016), GSM8K (Cobbe et al., 2021), AQUA (Ling et al., 2017), and SVAMP (Patel et al., 2021). 2) *symbolic reasoning:* Last Letters, Coin Flip (Wei et al., 2023), and Object Tracking, Bigbench Date. 3) *commonsense question answering:* CommonsenseQA (Talmor et al., 2019) and StrategyQA (Geva et al., 2021). For more details about the dataset please refer to Appendix A.

LLMs. To validate that Nash CoT is a general CoT method, we selected different large models as test models, including Mistral-7B (Instruct) (Jiang et al., 2023), GLM4-9B-Chat (Zeng et al., 2022; Du et al., 2022). In particular, all of these selected LLMs are turned via RL with human feedback (RLHF), and the difference between LLM turned with RLHF and the original foundation models have been detailed by Ouyang et al. (2022).

Baselines. The preliminary baselines we utilized include zero-shot, zero-shot CoT (Wei et al., 2023), andself-consistency (Wang et al., 2023). We test these approach with freezed LLMs.

Settings. Our evaluation on all selected tasks utilizes the same experimental settings bellow:

- **zero-shot and zero-shot CoT.** We follow the method proposed by Wei et al. (2023) and

Core LLM	Methods	Coin-Flipping	Last Letters	Object Tracking	Bigbench Date	Avg.
Mistral-Instruct (7B)	zero-shot	26.8 ± 5.1	0.0 ± 0.0	35.5 ± 4.1	31.1 ± 7.6	23.4± 14
	zero-shot CoT	27.9 ± 4.0	0.0 ± 0.0	30.1 ± 2.8	36.6 ± 5.4	23.6± 14
	self-consistency (20 Paths)	21.9± 4.7	0.0± 0.0	38.8± 0.8	47.0± 1.5	26.9± 18
	Nash CoT (10 Paths)	29.0± 5.4	0.5 ± 0.8	44.8 ± 2.0	41.1 ± 1.2	28.9± 17
GLM4-chat (9B)	zero-shot	27.3 ± 6.3	0.0 ± 0.0	38.8 ± 0.8	16.4 ± 2.3	22.2±4.7
	zero-shot CoT	87.4 ± 0.8	0.0 ± 0.0	37.7 ± 2.3	16.4 ± 4.8	41.1±22.9
	self-consistency (20 Paths)	98.9±1.5	0.0±0.0	37.7±2.3	16.4±4.8	44.0±26.0
	Nash CoT (10 Paths)	93.4±2.7	0.0 ± 0.0	37.7±2.3	16.4±4.8	42.4±24.4

Table 2: Experimental results on symbolic inference benchmarks. We test Zero-Shot CoT and Nash CoT with Mistral-Instruct (7B) and GLM4-chat (9B) on Symbolic QA benchmarks includes Coin-Flipping, Last Letters and Object Tracking. Among these baselines, Nash CoT performs the best.

Core LLM	Methods	StrategyQA	CommonsensQA	Avg.
Mistral-Instruct (7B)	zero-shot	49.2 ± 8.8	62.3 ± 4.8	55.8± 7
	zero-shot CoT	57.4±2.3	70.5 ± 2.7	64.0±7
	self-consistency (20 Paths)	59.6 ± 2.0	71.0 ± 3.4	65.3± 6
	Nash CoT (10 Paths)	56.8 ± 2.0	69.4 ± 4.7	63.1± 6
GLM4-chat (9B)	zero-shot	56.8 ± 4.7	17.5± 2.0	22.2±4.7
	zero-shot CoT	63.9±2.3	18.0±2.3	41.0± 22.9
	self-consistency (20 Paths)	69.9 ±3.3	18.0±2.3	44.0±26.0
	Nash CoT (10 Paths)	66.7 ± 0.8	18.0 ± 2.3	42.4±24.4

Table 3: Experimental results on Commonsense Reasoning. We test Zero-Shot CoT and Nash CoT with Mistral-Instruct (7B) and GLM4-chat (9B) on Commonsense Reasoning datasets includes StrategyQA and CommonsenseQA .

use the original template (e.g., "`Let's think step by step`") for evaluation.

- **self-consistency.** We follow Wang et al. (2023) to evaluate the performance ofself-consistency with selected LLMs, utilizing the zero-shot CoT template. Additionally, we set the number of inference paths to 20.

- **Nash CoT.** We set up n_{outer} as 3 and n_{mini} as 2, resulting in a total of $n_{\text{outer}} \times (n_{\text{mini}} + 1) + 1 = 10$ paths. Additionally, we have provided the player templates x^t in Table 1 and the Appendix, meanwhile we utilizing the same CoT template x^c as in zero-shot CoT.

Additionally, all evaluations are conducted on the inference of 60 random sampled questions multi times. And we have provided the mean and standard error in all tables.

5.1 Experimental Results

Evaluated Scores. The majority experimental results are demonstrated in table 1, 2 and 3. Nash CoT can improve Mistral-Instruct (7B) on almost all selected inference tasks, while showcasing similar performance to self-consistency with twice inference paths on GLM4-chat (9B). In particular, we have provided the total paths of Nash CoT that it only require the half of self-consistency, thus our claim in section 3 can be validated. When focusing on Mistral-Instruct (7B), Nash CoT has better performance on arithmetic and symbol inference tasks, showcasing its superiority performance on logic/math inference tasks. However, Nash CoT does not showcase improved performance in commonsense question answering tasks. We argue that this is because commonsense question answering tasks are more diverse, and the player template can't cover all topics. Therefore, the player template limits Nash CoT on commonsense question answering tasks. Importantly, we limit Nash CoT's

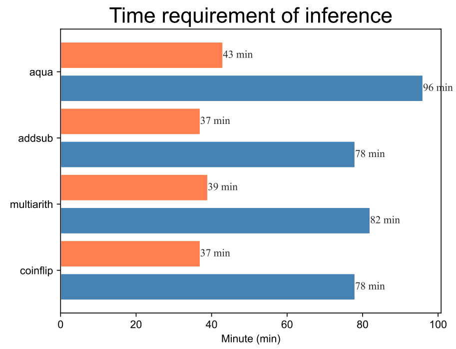

Figure 3: We used GLM4-chat (9B) on the same type of GPU (A-100) to evaluate Nash CoT and self-consistency across selected tasks. Nash CoT, employing a total of 10 paths, requires nearly half the time of self-consistency, which has 20 paths in total.

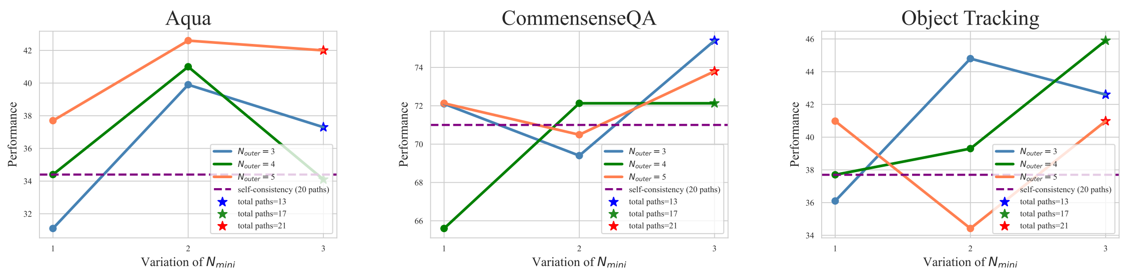

Figure 4: We use Mistral-Instruct (7B) to examine the impact of loop numbers on the inference performance of the large language model. Specifically, we used solid lines of specific colors to represent the experimental performance under certain N_{outer} as the N_{mini} changed. We marked self-consistency with 20 paths using dashed lines, and some results of Nash CoT, with total paths close to 20, were marked with stars.

performance by utilizing only total 10 paths for inference in this section. However, additional experimental results in the ablation section show that Nash CoT outperforms self-consistency via increasing the inference loops, thus Nash CoT can outperform self-consistency.

Inference Time. The path of Nash CoT are composed of three different kinds of types *i.e.* zero-shot CoT for problem inference (in loop N_{mini}), zero-shot CoT for player confining (in the outside loop of N_{outer}), and player template guided zero-shot CoT inference. Accordingly, different path requires different time. Therefore, we further count the total time requirement of self-consistency and Nash CoT in Figure 3, Nash CoT requires fewer inference time.

6 Ablation Study

In order to further validate the effectiveness of Nash CoT, we conducted extensive ablations to answer the following questions: 1) What will happen when the number of inference paths for Nash CoT is further increased? Will Nash CoT eventually surpass self-consistency, and what is the relationship between the number of loops and performance? 2) Does the template really improve the accuracy of path predictions, and what impact does it have on experimental performance?

As the number of inference paths increases, Nash CoT can obviously surpass self-consistency with fewer inference paths. To address question 1), we selected Mistral-Instruct (7B) and conducted evaluation on three different reasoning tasks, adjusting the N_{mini} and N_{outer}. As shown in Figure 4, as the number of loops increases, Nash CoT has a high probability of significantly outperforming self-consistency with fewer paths. However, different from self-consistency, the experimental results of Nash CoT do not show a monotonic (linear) relationship with the total number of total paths. This indicates that there is a significant difference between Nash CoT and self-consistency. Unlike Nash CoT, the experimental results of self-consistency show a clear improvement in performance as the number of paths increases.

GSM8K	AQua	SVAMP
55.7$\rightarrow$ 50.6	39.9 $\rightarrow$ 39.8	77.0 $\rightarrow$ 72.2

Table 4: Performance decreasing. We remove the mathematics from **Player Templates** and test Nash CoT on selected Arabic reasoning tasks.

The performance is impacted by the player template. To illustrate the impact of the template, we removed the mathematical templates from the **Player Templates** and then evaluated Nash CoT on selected Arabic reasoning. Results are shown in Table 6, showing an approximately 9.2% decrement in **GSM8K** and 6.2% decrement in **SVAMP**. Therefore, the performance of Nash CoT is impacted by the **Player Template**.

7 Conclusion

In this study, we proved the existence of Nash equilibrium in preference model, subsequently, we proposed a new CoT approach Nash CoT, and validated its performance on various inference benchmarks. Experimental results show that Nash CoT can perform equally or even better than self-consistency while only require half inference costs. In addition, we also conduct experiments to indicate that Nash CoT can also work on other benchmarks such as controllable text generation.

Limitations and Future Work

Despite Nash CoT showcase competitive performance with only half of inference paths, it requires pre-defined template, thus it's in-convenient to utilize Nash CoT in new emerging scenario, in the future we will develop a automatic approach to balance task feedback and template design.

Ethical Claims

Despite LLM has showcased superiority performance on vast benchmarks, but pre-train or fine-tune a LLM requires numerous computing resources. Therefore, it's crucial to study how to inference a LLM to reach the ceiling of its capacity. CoT is a ideal approach which has been proved that can obviously evaluate the performance of LLMs' inference. Among that,self-consistency is one of the best CoT approach.

Our method effectively reduce the inference times of multi-path inference, thereby reducing the deploy budgets of self-consistency. We believe our approach can further elevate the effectiveness of multi-path inference, thereby further improving the effectiveness of LLM.

References

Riad Akrour, Marc Schoenauer, and Michèle Sebag. 2011. Preference-based policy learning. In *ECML/PKDD*.

Tom Brown, Benjamin Mann, and etc. Ryder. 2020a. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020b. Language models are few-shot learners. *Preprint*, arXiv:2005.14165.

Róbert Istvan Busa-Fekete, Balázs Szörényi, Paul Weng, Weiwei Cheng, and Eyke Hüllermeier. 2013. Preference-based evolutionary direct policy search.

Weiwei Cheng, Johannes Fürnkranz, Eyke Hüllermeier, and Sang-Hyeun Park. 2011. Preference-based policy iteration: Leveraging preference learning for reinforcement learning. In *ECML/PKDD*.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. 2021. Training verifiers to solve math word problems. *Preprint*, arXiv:2110.14168.

Zhengxiao Du, Yujie Qian, Xiao Liu, Ming Ding, Jiezhong Qiu, Zhilin Yang, and Jie Tang. 2022. Glm: General language model pretraining with autoregressive blank infilling. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 320–335.

Hugo Touvron etc. 2023. Llama 2: Open foundation and fine-tuned chat models. *Preprint*, arXiv:2307.09288.

Mor Geva, Daniel Khashabi, Elad Segal, Tushar Khot, Dan Roth, and Jonathan Berant. 2021. Did aristotle use a laptop? a question answering benchmark with implicit reasoning strategies. *Preprint*, arXiv:2101.02235.

Mohammad Javad Hosseini, Hannaneh Hajishirzi, Oren Etzioni, and Nate Kushman. 2014. Learning to solve arithmetic word problems with verb categorization. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 523–533, Doha, Qatar. Association for Computational Linguistics.

Jiaxin Huang, Shixiang Shane Gu, Le Hou, Yuexin Wu, Xuezhi Wang, Hongkun Yu, and Jiawei Han. 2022. Large language models can self-improve. *Preprint*, arXiv:2210.11610.

Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. 2023. Mistral 7b. *Preprint*, arXiv:2310.06825.

Ziqi Jin and Wei Lu. 2023. Tab-cot: Zero-shot tabular chain of thought. *Preprint*, arXiv:2305.17812.

Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. 2023. Large language models are zero-shot reasoners. *Preprint*, arXiv:2205.11916.

Rik Koncel-Kedziorski, Hannaneh Hajishirzi, Ashish Sabharwal, Oren Etzioni, and Siena Dumas Ang. 2015. Parsing algebraic word problems into equations. *Transactions of the Association for Computational Linguistics*, 3:585–597.

Wang Ling, Dani Yogatama, Chris Dyer, and Phil Blunsom. 2017. Program induction by rationale generation : Learning to solve and explain algebraic word problems. *Preprint*, arXiv:1705.04146.

Rémi Munos, Michal Valko, Daniele Calandriello, Mohammad Gheshlaghi Azar, Mark Rowland, Zhaohan Daniel Guo, Yunhao Tang, Matthieu Geist, Thomas Mesnard, Andrea Michi, Marco Selvi, Sertan Girgin, Nikola Momchev, Olivier Bachem, Daniel J. Mankowitz, Doina Precup, and Bilal Piot. 2023. Nash learning from human feedback. *Preprint*, arXiv:2312.00886.

OpenAI. 2024. Gpt-4 technical report. *Preprint*, arXiv:2303.08774.

Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Christiano, Jan Leike, and Ryan Lowe. 2022. Training language models to follow instructions with human feedback. *Preprint*, arXiv:2203.02155.

Arkil Patel, Satwik Bhattamishra, and Navin Goyal. 2021. Are NLP models really able to solve simple math word problems? In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2080–2094, Online. Association for Computational Linguistics.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners.

Jack W. Rae, Sebastian Borgeaud, Trevor Cai, Katie Millican, Jordan Hoffmann, Francis Song, John Aslanides, Sarah Henderson, Roman Ring, Susannah Young, Eliza Rutherford, Tom Hennigan, Jacob Menick, Albin Cassirer, Richard Powell, George van den Driessche, Lisa Anne Hendricks, Maribeth Rauh, Po-Sen Huang, Amelia Glaese, Johannes Welbl, Sumanth Dathathri, Saffron Huang, Jonathan Uesato, John Mellor, Irina Higgins, Antonia Creswell, Nat McAleese, Amy Wu, Erich Elsen, Siddhant Jayakumar, Elena Buchatskaya, David Budden, Esme Sutherland, Karen Simonyan, Michela Paganini, Laurent Sifre, Lena Martens, Xiang Lorraine Li, Adhiguna Kuncoro, Aida Nematzadeh, Elena Gribovskaya, Domenic Donato, Angeliki Lazaridou, Arthur Mensch, Jean-Baptiste Lespiau, Maria Tsimpoukelli, Nikolai Grigorev, Doug Fritz, Thibault Sottiaux, Mantas Pajarskas, Toby Pohlen, Zhitao Gong, Daniel Toyama, Cyprien de Masson d'Autume, Yujia Li, Tayfun Terzi, Vladimir Mikulik, Igor Babuschkin, Aidan Clark, Diego de Las Casas, Aurelia Guy, Chris Jones, James Bradbury, Matthew Johnson, Blake Hechtman, Laura Weidinger, Iason Gabriel, William Isaac, Ed Lockhart, Simon Osindero, Laura Rimell, Chris Dyer, Oriol Vinyals, Kareem Ayoub, Jeff Stanway, Lorrayne Bennett, Demis Hassabis, Koray Kavukcuoglu, and Geoffrey Irving. 2022. Scaling language models: Methods, analysis insights from training gopher. *Preprint*, arXiv:2112.11446.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Stefano Ermon, Christopher D. Manning, and Chelsea Finn. 2023. Direct preference optimization: Your language model is secretly a reward model. *Preprint*, arXiv:2305.18290.

Subhro Roy and Dan Roth. 2016. Solving general arithmetic word problems. *Preprint*, arXiv:1608.01413.

Freda Shi, Mirac Suzgun, Markus Freitag, Xuezhi Wang, Suraj Srivats, Soroush Vosoughi, Hyung Won Chung, Yi Tay, Sebastian Ruder, Denny Zhou, Dipanjan Das, and Jason Wei. 2022. Language models are multilingual chain-of-thought reasoners. *Preprint*, arXiv:2210.03057.

Alon Talmor, Jonathan Herzig, Nicholas Lourie, and Jonathan Berant. 2019. Commonsenseqa: A question answering challenge targeting commonsense knowledge. *Preprint*, arXiv:1811.00937.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2023. Self-consistency improves chain of thought reasoning in language models. *Preprint*, arXiv:2203.11171.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. 2023. Chain-of-thought prompting elicits reasoning in large language models. *Preprint*, arXiv:2201.11903.

Aaron Wilson, Alan Fern, and Prasad Tadepalli. 2012. A bayesian approach for policy learning from trajectory preference queries. In *Neural Information Processing Systems*.

Aohan Zeng, Xiao Liu, Zhengxiao Du, Zihan Wang, Hanyu Lai, Ming Ding, Zhuoyi Yang, Yifan Xu, Wendi Zheng, Xiao Xia, et al. 2022. Glm-130b: An open bilingual pre-trained model. *arXiv preprint arXiv:2210.02414*.

Zhuosheng Zhang, Aston Zhang, Mu Li, and Alex Smola. 2022. Automatic chain of thought prompting in large language models. *Preprint*, arXiv:2210.03493.

A Dataset

Our majority dataset are composed of three different kinds of inference tasks. 1) *arithmetic reasoning:* SingleEq (Koncel-Kedziorski et al., 2015), AddSub (Hosseini et al., 2014), MultiArith (Roy and Roth, 2016), GSM8K (Cobbe et al., 2021), AQUA (Ling et al., 2017), and SVAMP (Patel et al., 2021). 2) *symbolic reasoning:* Last Letters, Coin Flip (Wei et al., 2023), and Object Tracking, Bigbench Date. 3) *commonsense question answering:* CommonsenseQA (Talmor et al., 2019) and StrategyQA (Geva et al., 2021). For more details about the dataset please refer to (Wang et al., 2023).

B Uesage of LLM.

We utilize LLM to rectify grammar errors.

C Computing Resources

Our experiments were run on a computer cluster with 32GB RAM, 4-Core CPU, and NVIDIA-A100 (80G, 32G)/NVIDIA-V100 (32G) GPU, Linux platform.

D Source Code.

We have provided source code for reference. Additionally, our code are based on `https://github.com/amazon-science/auto-cot` and refer to the coding manner from `https://github.com/eureka-research/Eureka`.

E Proof of theorem 3.1.

Subsequently, we prove the existence of Nash equilibrium in this system. For any two given polices $\pi_1 \in \Pi$ and $\pi_2 \in \Pi$ We first define the pay-off of π_1 and π_2 as $R(\pi_1;\pi_2)$ and $R(\pi_2;\pi_1)$:

$$\begin{aligned} R(\pi_1;\pi_2) &= \mathcal{P}(\pi_1 \prec \pi_2) \\ R(\pi_2;\pi_1) &= \mathcal{P}(\pi_1 \succ \pi_2), \end{aligned} \tag{3}$$

we provide the proof of the existence of Nash equilibrium in this system. We define $\bar{\pi} = [\pi_1, \pi_2]$, $v(\bar{\pi}) = [R(\pi_1;\pi_2), R(\pi_1;\pi_2)]$. According to the Nash equilibrium, it should have to satisfy this relationship:

$$v(\bar{\pi}^*)(\bar{\pi}^* - \bar{\pi}) \leq 0 \tag{4}$$

Subsequently, refer to , we can learn that if we want Equation 4 holds true, we just have to guarantee Equation 5 holds true.

$$(v(\bar{\pi}) - v(\bar{\pi}'))^{\mathrm{T}}(\bar{\pi} - \bar{\pi}') \leq 0, \tag{5}$$

where $\bar{\pi}$ and $\bar{\pi}'$ are any two given policy set. Subsequently, we can further darrive at the following relationships:

$$
\begin{aligned}
(v(\bar{\pi}) - v(\bar{\pi}'))^{\mathrm{T}}(\bar{\pi} - \bar{\pi}') &= \begin{pmatrix} R(\pi_1;\pi_2) - R(\pi_1';\pi_2') \\ R(\pi_2;\pi_1) - R(\pi_2';\pi_1') \end{pmatrix} \cdot (\pi_1 - \pi_1', \pi_2 - \pi_2') \\
&= \Big(R(\pi_1;\pi_2) - R(\pi_1';\pi_2') \Big) \cdot (\pi_1 - \pi_1') + \Big(R(\pi_2;\pi_1) - R(\pi_2';\pi_1') \Big) \cdot (\pi_2 - \pi_2') \\
&= \Big(\mathcal{P}(\pi_1 \prec \pi_2) - \mathcal{P}(\pi_1' \prec \pi_2') \Big) \cdot (\pi_1 - \pi_1') + \Big\{ 2 - \Big(\mathcal{P}(\pi_1 \prec \pi_2) + \mathcal{P}(\pi_1' \prec \pi_2') \Big) \Big\} \\
&\cdot (\pi_2 - \pi_2') \\
&= \Big(\mathcal{P}(\pi_1 \prec \pi_2) - \mathcal{P}(\pi_1' \prec \pi_2') \Big) \cdot (\pi_1 - \pi_1' - \pi_2 + \pi_2') + 2 \cdot (\pi_2 - \pi_2') \\
&= \Big(\mathcal{P}(\pi_1 \prec \pi_2) - \mathcal{P}(\pi_1' \prec \pi_2') - 2 \Big) \cdot (\pi_2' - \pi_2) + \\
&\Big(\mathcal{P}(\pi_1 \prec \pi_2) - \mathcal{P}(\pi_1' \prec \pi_2') \Big) \cdot (\pi_1 - \pi_1').
\end{aligned}
\tag{6}
$$

In particular, we can find that if $\bar{\pi} \equiv \bar{\pi}'$ then $(v(\bar{\pi}) - v(\bar{\pi}'))^{\mathrm{T}}(\bar{\pi} - \bar{\pi}') \equiv 0$, thus $\bar{\pi} \equiv \bar{\pi}'$ is one solution that π_1 and π_2 has reached equilibrium.