Plan and Solve in approaching AI

Created with **Gemini** February 13, 2025 at 02:52 PM Published February 13, 2025 at 03:11 PM

What I intend to do is simple: I want to solve a problem with the helping hand of an AI language model.

Now, in order to 'solve' a problem I have to have a '...' how to do, right?

Well, according to you, what do I have to have?

To solve a problem with the help of an AI language model, you need to have a **clearly defined problem** and a **strategy or plan** for how to approach it. The more specific and well-structured these are, the better the AI can assist you.

Here's a breakdown of what's helpful:

- **Clearly Defined Problem (well-defined problem statement):** Vague problems lead to vague solutions. Break down your problem into smaller, more manageable parts. Be specific about what you want to achieve. For example, instead of "improve my writing," try "improve the clarity and conciseness of the introduction to my essay on climate change."

- **Strategy/Plan:** This doesn't have to be overly formal, but you should have some idea of how you want to tackle the problem. This might involve:

    - **Specific Questions:** What questions do you need to answer to solve the problem?
    - **Steps:** What are the steps involved in finding a solution?
    - **Information Needs:** What information do you need to gather?
    - **Expected Outcome:** What does a successful solution look like?

- **Understanding of the AI's Capabilities (and Limitations):** AI language models are powerful, but they're not magic. They're good at generating text, translating languages, writing different kinds of creative content, and answering your questions in an informative way. They are not good at providing factual information that requires real-world experience, or making subjective judgements. You need to be aware of what the AI can and cannot do to effectively use it.

- **Iteration and Refinement:** Don't expect the AI to give you the perfect solution on the first try. Be prepared to iterate, refine your prompts, and adjust your strategy based on the AI's responses. Think of it as a collaborative process.

In short, you need a *plan of attack* for your problem, and a good understanding of how the AI can be a *tool* in that attack. The clearer your problem and strategy, the more effectively you can use the AI's capabilities.

To solve a problem with the help of an AI language model, I need to have a clearly defined problem. For such a problem to be solved I need a strategy or plan for how to approach it, you tell me.

A 'strategy' would be superb, I guess, but even if I don't have one, I can have a plan. A plan in order to get the problem solved, right?

Yes, a plan is definitely essential, and it's often the first step *towards* developing a full-fledged strategy. Think of it this way: a plan is like the initial sketch, and the strategy is the more detailed, refined painting.

You can start with a plan even if you don't have a complete strategy. A good plan outlines the *steps* you think you need to take to solve the problem. It might not be perfect, and it might change as you go, but it gives you a starting point.

Here's why a plan is so important, especially when working with an AI:

- **Focus:** A plan helps you focus your interaction with the AI. Instead of just throwing questions at it randomly, you have a specific direction.
- **Iteration:** A plan allows for iteration. As you work through your plan, you'll likely get new information or realize that some steps need to be adjusted. That's perfectly normal, and the AI can help you with that refinement.
- **Evaluation:** A plan makes it easier to evaluate the AI's output. You can compare the AI's suggestions to your plan and see how well they align. This helps you determine if the AI is being helpful and if you're on the right track.
- **Prompt Engineering:** A plan helps you formulate better prompts for the AI. Instead of asking vague questions, you can ask specific questions related to the steps in your plan. This leads to more useful and targeted responses from the AI.

So, even if you don't have a fully formed strategy, start with a plan. List the steps you think are necessary to solve your problem. Then, use the AI to help you refine those steps, explore different approaches, and develop a more comprehensive strategy. The plan is your roadmap, and the AI can be your guide.

Now, what do you think about the plan-and-solve prompting strategy.

A sound plan-to-solve approach in prompting an AI-model. Would you consider such an approach a strategy?

The "plan-and-solve" prompting strategy is absolutely a valid and often highly effective approach, and yes, I would consider it a *strategy* in itself, or at least a core component of a larger strategy. It's a structured way of interacting with the AI that leverages its strengths in processing information and generating text.

Here's why the plan-and-solve approach is so powerful:

- **Decomposition:** It encourages you to break down a complex problem into smaller, more manageable sub-problems (the "plan" part).

  This is crucial for both problem-solving in general and for working effectively with AIs. AIs, while powerful, benefit from being given focused tasks.

- **Structured Interaction:** It provides a framework for your interaction with the AI. You're not just asking random questions; you're guiding the AI through a logical process.

  This makes the interaction more productive and helps you get more relevant and useful responses.

- **Iterative Refinement:** The plan-and-solve approach naturally lends itself to iteration. You can use the AI's output to refine your plan, identify gaps in your understanding, or explore alternative solutions.

  This iterative process is key to successful problem-solving.

- **Improved Prompting:** Having a plan allows you to craft more specific and targeted prompts. Instead of asking the AI to solve the entire problem at once, you can ask it to help you with specific steps in your plan.

  This leads to more focused and helpful responses.

- **Mimicking Human Thought:** The plan-and-solve approach mirrors how humans often tackle complex problems. We don't usually jump straight to a solution; we develop a plan, consider different options, and then execute the plan.

By using this approach with an AI, you're leveraging its ability to augment your own thinking process.

So, while "plan-and-solve" might be a *component* of a more comprehensive strategy (which could also include things like research, experimentation, and evaluation), it's definitely a strategy *in its own right*, and *a very effective one* for using AI to solve problems. It's a structured approach to prompting that brings significant benefits.

One last thing:

Another LLM tells me: [quote] Plan-and-Solve (PS) Prompting is just one approach within the broader Chain-of-Thought (CoT) reasoning framework. Various methods, like Least-to-Most (LtM), Tree-of-Thought (ToT), and Program-of-Thought (PoT) prompting, all aim to improve AI reasoning by structuring its thought process [/quote]

Do you fully agree with the quote?

If so, could you clarify the pros and cons of cons of PS against LtM, ToT and PoT?

I agree with the quote. Plan-and-Solve (PS) prompting *is* related to and can be considered a simpler form of Chain-of-Thought (CoT) prompting.

The other methods mentioned (LtM, ToT, and PoT) build upon the basic idea of structuring the AI's reasoning process in more sophisticated ways.

Here's a breakdown of the pros and cons of PS compared to the others:

Plan-and-Solve (PS):

- **Pros:**
    - **Simplicity:** Easier to implement and understand. It's a good starting point for exploring structured prompting.
    - **Directness:** Focuses on creating a plan and then executing it, which is a natural way to approach many problems.
    - **Efficiency:** Can be more efficient than more complex methods for problems that don't require extensive exploration of different paths.
- **Cons:**
    - **Limited Exploration:** May not be as effective for problems that require exploring multiple possibilities or dealing with significant uncertainty. It's less flexible than ToT.

    - **Less Robust:** Can be more susceptible to errors if the initial plan is flawed. It doesn't have the built-in backtracking or alternative path exploration of ToT.
    - **Less Suited for Complex Reasoning:** May struggle with problems that require deep logical reasoning or multiple steps of inference.

Least-to-Most (LtM):

- **Pros:**
    - **Structured Approach:** Breaks down problems into subproblems and solves them in a specific order (from least to most complex). This can be very helpful for complex problems.
- **Cons:**
    - **Rigid Structure:** The strict ordering of subproblems can be limiting. It might not be suitable for problems where the dependencies between subproblems are less clear.
    - **Requires Careful Decomposition:** The success of LtM depends heavily on how well the problem is decomposed.

Tree-of-Thought (ToT):

- **Pros:**
    - **Exploration of Multiple Paths:** Allows the AI to explore multiple reasoning paths and evaluate them. This is very effective for complex problems with uncertainty.
    - **Backtracking and Refinement:** Can backtrack and revise previous decisions based on new information.
    - **Robustness:** More robust than PS because it considers multiple options.
- **Cons:**
    - **Complexity:** More complex to implement than PS. Requires careful management of the tree structure.
    - **Computational Cost:** Can be computationally expensive, as it involves evaluating multiple paths.

Program-of-Thought (PoT):

- **Pros:**
    - **Leverages Code Generation:** Uses code to represent and execute reasoning steps. This can be very powerful for problems that involve computation or symbolic reasoning.
    - **Formal and Precise:** Code provides a formal and precise way to represent knowledge and reasoning steps.

- **Cons:**
  - **Requires Programming Skills:** Requires some level of programming knowledge to implement.
  - **Limited Applicability:** Not suitable for all types of problems. Best suited for problems that can be naturally expressed in code.

In summary:

PS is a good starting point for structured prompting.
LtM is useful for systematically solving complex problems.
ToT is more robust and allows for exploration of multiple paths.
PoT is powerful for problems that can be expressed in code.
The best approach depends on the specific problem you are trying to solve.