Algorithm for conceptual problem-solving using General Environment Description Language

Nina Bąkowska[1,*,†] and Krzysztof Zatwarnicki[1,†]

[1] *Opole University of Technology, Proszkowska 76, 45-758 Opole, Poland*

Abstract

The development of automated systems has been demonstrated to be a significant area of study, with rapid advancement occurring with the advent of artificial intelligence. In order to provide a universal method of problem-solving, an algorithm for executing various tasks within environments described with the General Environment Description Language (GEDL) has been developed. The initial version of the algorithm, which operated at a conceptual level, was confronted with considerable challenges, including inefficiency in the utilization of resources, the potential for infinite loops, and the difficulty of applying conceptual solutions to real-world scenarios. To address these issues, several key enhancements were introduced. These included the implementation of a state queue to prevent repetitive state exploration and optimization in the management of instances, features, and relationships. The enhanced algorithm was evaluated using a practical case study involving a washing machine, wherein it effectively circumvented infinite loops and markedly reduced execution time. Additionally, the algorithm was capable of identifying absent instances and relationships. The outcomes illustrate that the modifications not only enhance the algorithm's performance but also expand its applicability to more complex and diverse environments. Future research will concentrate on integrating evolutionary algorithms to further optimize solutions and address challenges related to missing instances and implicit parameters in the problem-solving process.

Keywords

automated systems, algorithm, algorithm optimization, environment description languages, problem solving, knowledge representation, problem-solving

1. Introduction

The development of automated systems capable of performing tasks without human supervision has witnessed considerable advancement in recent years. From industrial robots to intelligent personal assistants, these systems have had a profound impact on numerous facets of daily life and industry [1]. This drive towards automation can be seen as a fundamental aspect of human nature, namely the tendency to solve problems. This has been a significant factor in the advancement of technology over time.

ITTAP'2024: 4th International Workshop on Information Technologies: Theoretical and Applied Problems, November 20–22, 2024, Ternopil, Ukraine, Opole, Poland * Corresponding author.
† These authors contributed equally.

bakowskanina1704@gmail.com (N. Bąkowska); k.zatwarnicki@po.opole.pl (K. Zatwarnicki); 0009-0000-7719-2379 (N. Bąkowska); 0000-0001-6156-6030 (K. Zatwarnicki)

The introduction of artificial intelligence (AI) marked a significant turning point in the evolution of automated systems. Artificial intelligence has developed rapidly, offering sophisticated solutions to complex problems across a wide range of domains. Nevertheless, while AI displays considerable proficiency in certain domains, there is a clear requirement for more adaptable systems that can accommodate a diverse range of environments and challenges, rather than being constrained by a narrow focus on a single application.

In response to this need, our previous work [2] has concentrated on the creation of the General Environment Description Language (GEDL) [3] and an accompanying algorithm designed to resolve issues in environments described by GEDL. The objective of this approach was to provide a more general solution framework that operates at a conceptual level. The algorithm was structured in a manner that enabled navigation through a series of abstract actions and states, thereby effectively solving problems in a simulated, idealised environment. Although this conceptual approach proved valuable in generating solutions that could theoretically achieve the desired goals, it became evident that the algorithm encountered difficulties when applied to real-world scenarios.

The fundamental challenge lies in the fact that the algorithm, by design, operates primarily at the conceptual level, where the inherent complexity of real-world instances— such as varying parameters, unpredictable conditions, and multiple potential instances of objects—is not fully accounted for. The conceptual solutions generated by the algorithm do not automatically translate into executable actions within a real environment. This is due to the necessity of considering physical constraints, resource availability and the precise state of objects, which are absent from the conceptual level. Consequently, while the algorithm effectively delineates a potential sequence of actions in an idealised environment, it frequently fails to address the practical challenges encountered when those actions are mapped onto real-world occurrences.

This discrepancy between conceptual problem-solving and practical execution demonstrates the shortcomings of the original algorithm. It is crucial to overcome the gap between abstract planning and its real-world implementation. It needs to be ensured that solutions are not only theoretically sound but also executable in the actual environments in which they are intended to operate.

2.Related work

This research is aligned with the broader goals of artificial general intelligence (AGI) [4], which aims to create intelligent systems capable of performing any intellectual task that a human can undertake. The ability to solve problems is a fundamental aspect of human cognition. It involves the organisation and structuring of knowledge in a flexible and adaptive manner, which is a key area of focus in the development of AGI. Research in this field has contributed to the theoretical foundations of AGI by exploring how human-like general intelligence can be structured within AI architectures. This emphasises the importance of knowledge management and reasoning.

A number of description languages have been developed with the objective of formalising and organising a variety of tasks and domains within artificial environments. Such languages include Game Description Languages (GDL) [5], which are used to describe the rules and dynamics of different games, and Video Game Description Languages (VGDL) [6], which extend this concept to video games. Although both GDL and VGDL have proven useful in specific contexts, their primary drawback is their lack of universality, as they are designed exclusively for use with games. However, languages such as GDL-III [7] have introduced the ability to model more complex scenarios by incorporating epistemic logic, which accounts for agents' knowledge within the game environment.

In contrast, agent-based description languages, such as the JIAC Agent Description Language (JADL) [8] and the Java Agent Development Framework Language (JADEL) [9], are more versatile in their scope, focusing on defining the characteristics and behaviours of agents within multi-agent environments. These languages provide frameworks for modelling interactions between multiple agents, thereby enabling the design of more sophisticated simulations that extend beyond rigid rule-based systems. Moreover, the research on compositional languages in multi-agent populations introduces a grounded approach, whereby agents develop their language to coordinate actions and behaviours within an environment.

Our work is based on the General Environment Description Language (GEDL), which provides an even broader framework for representing complex environments and interactions. In contrast to the aforementioned languages, GEDL offers a wide range of possibilities for describing dynamic and diverse environments. The flexibility of GEDL makes it particularly suitable for this research, where complex problem-solving requires a detailed representation of both the environment and the relationships between its entities.

A systematic approach to provide a reliable solution has resulted in the development of an initial algorithm designed to solve problems within environments defined by the GEDL. This earlier approach established the foundation for the derivation of fundamental conceptual solutions within a simulated environment. However, it also revealed significant challenges, including inefficiencies in resource management, the potential for infinite loops, and the difficulty of applying conceptual solutions to real-world occurrences. These insights have informed the development of the enhanced algorithm presented in this paper, which addresses these limitations and extends the applicability of GEDL to more complex and realistic environments.

In recent years, there has been a notable increase in interest in the areas of task execution and problem-solving within the context of specific simulated environments, such as the video game Minecraft. The frameworks MCU [10] and JARVIS-1 [11] represent significant advances in the evaluation of multi-task agents and agent-based control in openworld environments. To illustrate, MCU introduces a task-centric framework designed to evaluate agent performance across a wide variety of tasks, employing a set of difficulty metrics to assess an agent's capabilities. Similarly, JARVIS-1 employs multi-modal language models and memory-augmented systems to facilitate the completion of complex tasks within the open-ended Minecraft environment. These systems offer valuable insights into problem-solving and task planning in environments that exhibit a high degree of complexity and variability.

Another notable advancement in AGI development is evidenced by ChatGPT [12], which is capable of generating step-by-step problem-solving solutions. However, ChatGPT's reliance on natural language processing and statistical models limits its reliability in complex environments that require precise control over the problem-solving process. By leveraging GEDL, our work offers a more structured control over the environment and greater transparency into the steps required to solve problems in dynamic, multi-agent settings.

2.1. Motivation

The objective of this work is to present an enhanced approach that addresses the issues previously identified, thereby improving the reliability and efficiency of problem-solving in complex environments. It is hoped that, by confronting a previously developed approach with an example that highlights its shortcomings, the importance of readjusting the solution to cover a wider range of problems will become clear. This marks a major advance in enhancing the effectiveness of existing algorithm and opening up new possibilities for their use. The objective of this paper is to provide a more robust and flexible solution that can handle the details of complex environments, leading to the acquisition of more reliable and efficient outcomes.

3. General Environment Description Language

The General Environment Description Language is a framework designed to provide a description of the environment in which robots or agents operate. This enables the assignment of meaning to objects and the planning of tasks based on those objects. The GEDL is inspired by the manner in which the human mind organises and processes knowledge. The human mind constructs a conceptual system, a structured set of ideas and thoughts, and learns from mistakes, thereby refining this system over time. Furthermore, humans are capable of passing entire conceptual frameworks to others, thereby reducing the time and resources that would otherwise be expended on individual learning.

In GEDL, the term "environment" is used to represent a physical or conceptual reality, comprising individuals and objects. The acquisition of knowledge is achieved through the cognitive mechanisms that enable individuals to recognise objects as instances with specific features. These features are grouped into sets that are associated with particular instances. The relationships between objects or individuals are based on logical connections derived from the aforementioned features. Furthermore, the language defines the concept of "relationship" which categorises these connections.

In GEDL, instances are capable of performing actions, either independently or with assistance, with the objective of modifying the environment. Actions modify the state of the environment by altering the features, relationships, or even the existence of objects. Each action is regarded as an indivisible unit; for instance, the placement of an object on a shelf is treated as a single, complete action, rather than a series of smaller movements.

A problem in GEDL is defined as the task of modifying the environment from its initial state to a desired final state, utilising the available actions. In some cases, multiple solutions may exist, with the optimal one depending on the desired outcome, such as the minimisation of time or steps.

The formation of an individual's understanding of reality is a function of accumulated knowledge, which is referred to as "individual knowledge." This includes information collected from the environment and concepts learned. This knowledge can be developed through observation, deduction, and the exchange of information with others, the latter of which has been demonstrated to be an effective method for accelerating learning. In GEDL, individual knowledge is comprised of three elements:

- Conceptual System - represents the individual's perception of entities, relationships, and activities. This represents the individual's perception of entities, relationships, and activities. It encompasses features, feature sets, instance concepts, relationship concepts, and action concepts. A feature is defined by a name and a range of possible values, which allows instances to be classified. The grouping of attributes that are common to objects of the same instance constitutes a feature set, thereby facilitating the process of classification. Relationship concepts define the connections between instances, which may be subject to alteration through the performance of actions.
- Occurrences - these are particular instances and relationships that are identified by the individual. In contrast to the broader Conceptual System, Occurrences are concrete entities, characterised by defined features and values that can be acted upon.
- Experience - this aspect of knowledge provides the impetus for the search for solutions to problems. By defining a problem in terms of the current and desired state of the environment, the individual employs their experiences to identify pertinent actions and construct a step-by-step plan to achieve the desired outcome.

GEDL offers a systematic methodology for describing environments, comprehending the relationships between objects, and planning effective actions to address problems. It draws heavily upon the principles of human cognition, including how we learn, organize, and share knowledge.

3.1. Example

In order to more clearly demonstrate the potential of GEDL notation, a familiar, everyday task – namely, the washing of clothes – was described using this notation. Although the act of laundering clothing may appear to be a relatively simple and straightforward process, it is, in fact, comprised of a series of complex steps and requires the fulfilment of several prerequisites before any action can be initiated. The environment is defined using JSON syntax. This allows the description to be both universal and easily understood by humans.

In this scenario, the Conceptual System comprises several key concepts of components, including a manipulator, a washing machine, a laundry basket, and the laundry itself. Each of these components is represented as a distinct instance concept within the GEDL framework, complete with specific features and relationships pertinent to the task at hand. To illustrate, the washing machine concept includes features such as 'turned on', 'contains laundry', 'contains detergent', 'washing', 'selected washing program', 'access inside' and 'access drawer'. These features are directly related to the washing machine's functions; irrelevant attributes such as size, power consumption and water usage are omitted since they do not impact the task at hand. A comparable level of detail is applied to the remaining components. The manipulator is assigned a feature indicating whether it is holding an object, the basket is assigned a feature indicating whether it contains laundry, and the laundry is assigned a feature indicating whether it is clean. Furthermore, two relationship concepts, "holds" and "contains," are defined to specify the interactions between these objects, such as the manipulator holding an item or the basket containing laundry. The simplified example illustrated in Figure 1 provides a helpful overview of this concept. It should be acknowledged that this illustration do not provide a comprehensive account of the details in question; rather, its purpose is to assist the reader in understanding the manner in which the Conceptual System is represented.

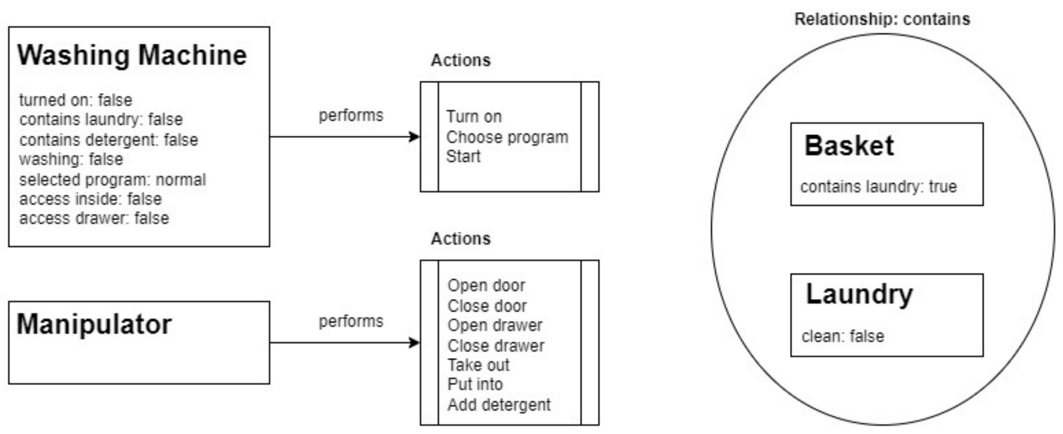

Figure 1 Simplified illustration of the doing laundry example

To represent the entire process of doing laundry, twelve distinct action concepts are defined. These include the following: 'takeOutOfBasket', 'openDoor', 'closeDoor', 'putIntoWashingMachine', 'openDrawer', 'closeDrawer', 'addDetergent', 'turnOnWashingMachine', 'chooseProgram', 'pressStart', 'takeOutOfWashingMachine', and 'putInBasket'. It should be noted that the sequence in which the actions are performed does not necessarily correspond with that of a one-to-one set of steps. In some instances, the performance of an action may necessitate the repetition of a previous action in order to complete the task; this reflects the iterative nature of numerous activities that occur on a daily basis.

Each action is associated with specific objects in the environment and is subject to certain conditions. To illustrate, the action of placing laundry into the washing machine necessitates the presence of three objects in designated states: the manipulator must be in possession of dirty laundry, and the washing machine must be in an open and unloaded configuration. Once the action has been completed, the manipulator is no longer holding any items and the laundry is inside the washing machine. However, the door remains open since closing it is treated as a separate atomic action.

The Occurrences section delineates the initial state of the environment at the inception of the laundry task. In this initial state, the manipulator is not holding any items, the basket contains dirty laundry, and the washing machine is turned off, empty, and without detergent, with no access inside. This part mirrors the state of actual, existing instances.

The problem, as defined in the Experience section, is to successfully complete the laundry task using the instances defined in the individual's knowledge, which include the manipulator, washing machine, basket, and laundry. The ultimate goal is to obtain clean clothes. The only difference between the initial and final state is the value of the 'clean' feature. However, achieving this goal is not a matter of performing a single action; rather, it requires a series of modifications to the instances described in Occurrences.

4. Initial solution

The primary objective of the algorithm remains unchanged across all existing versions. It is to determine a path from an initial state to a final state by methodically exploring all possible actions that can be executed in a given environment. The approach involves a systematic exploration of possible state transitions to ascertain whether a solution exists and to outline the steps required to achieve the desired final state.

Initially, the algorithm takes the given initial state and evaluates the set of possible actions that can be performed within that state. Each feasible action leads to the creation of a new state, which is then subjected to a similar evaluation process. This iterative process continues, with the algorithm exploring each newly generated state by determining the subsequent actions that can be performed.

The process begins with the algorithm analyzing the prevailing environmental circumstances, taking into account initial positions, features, conditions and any existing relationships between objects. The data is then employed in the construction of a simulated scenario, which is utilized as a means of facilitating the process of problem-solving.

After that, the algorithm identifies and executes a feasible action based on the current state. This approach enables the determination of the specific (on the conceptual level) instance that performs the action at any given time. Once the action has been selected, it is executed, resulting in alterations to the environment. These may include the movement of objects, the modification of their properties, or the creation of new relationships.

Subsequently, the algorithm determines whether the objective has been reached. If this is the case, the sequence of actions is identified as a potential solution. Conversely, if the objective has not been met, the algorithm commences the process anew from the updated environmental state, searching for and executing the next available action. The algorithm's execution terminates under two conditions: if the final state is achieved, indicating that the problem is solved, or if there are no more states left to explore, signifying that no solution exists for the given problem. The illustration of this process is presented in the Figure 2.

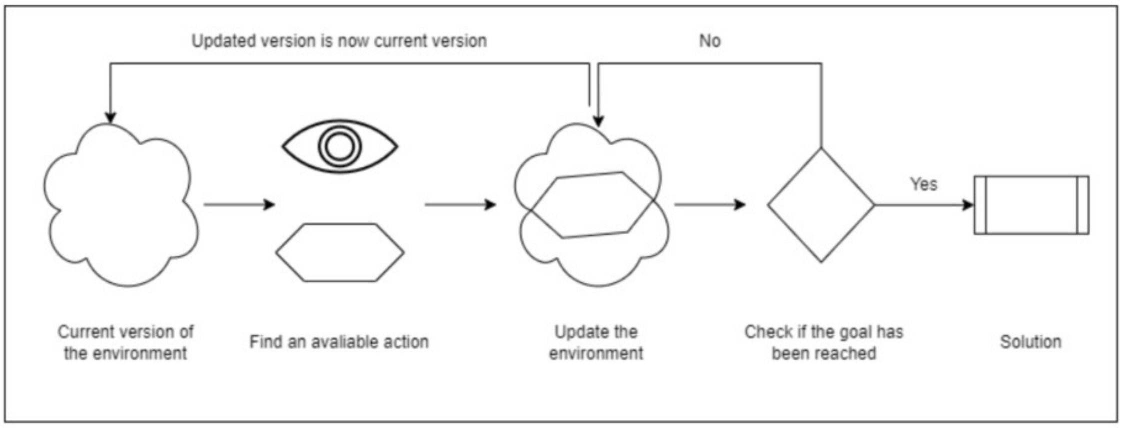
Updated version is now current version
No
Yes
Current version of the environment
Find an avaliable action
Update the environment
Check if the goal has been reached
Solution

Figure 2: Illustrated process of finding a solution

4.1. Challenging example

The task of doing laundry can be used as a practical example to illustrate the challenges encountered in early stages of algorithmic design. The conceptual problem is straightforward: the initial state features dirty laundry, and the final goal is to have clean laundry. In order to achieve this goal, a series of actions must be carried out. These include removing the laundry from the basket, opening the washing machine door, placing the laundry inside, closing the door, adding detergent, starting the machine, waiting for the wash cycle to complete, removing the laundry, and finally, placing it back in the basket.

Although the conceptual solution may appear straightforward, several issues emerge when considering the possibility of multiple outcomes or translating it into a practical algorithm that operates within a real environment. For the sake of this example, we may consider the following scenario: the environment consists of multiple instances, including two washing machines (one functional and one broken), two baskets (one with laundry and one empty), a piece of dirty laundry, and a manipulator (a mechanical arm) that can interact with these objects.

4.1.1.Time and Resource Consumption

A significant challenge is the vast number of potential outcomes that must be taken into account at each stage of the process. In any given state, the manipulator is capable of performing a range of actions, including opening doors, moving laundry, and adding detergent, among others. Each of these actions results in a new state, and all potential outcomes must be considered to guarantee that the algorithm does not overlook a viable path to the goal. The combinatorial explosion of potential actions and states renders the algorithm highly resource-intensive, as it must explore a multitude of potential sequences of actions. The necessity to assess each of these possibilities results in a considerable investment of time and resources, which has the potential to significantly impair the efficiency of the algorithm.

4.1.2.Looping

A further significant issue is that of infinite loops, which is particularly evident in the context of the laundry example. The environment permits actions that are cyclical in nature, such as repeatedly opening and closing the washing machine door or the detergent drawer. Moreover, actions such as placing laundry into the washing machine, closing the door, opening the door, removing the laundry, and then repeating the process could result in an infinite loop. Such cycles are not conducive to achieving the desired outcome; rather, they result in the inefficient utilization of computational resources and time. Without careful management, the likelihood of the algorithm becoming trapped in these futile loops is high. This would effectively halt its ability to progress towards the final state.

4.1.3.Real World Application

A final and equally crucial challenge is to ascertain whether the conceptual solution can be implemented in the actual environment. To illustrate, the conceptual solution is based on the assumption that a functional washing machine is available. However, should the only available washing machine be in a state of malfunction, the entire plan becomes unworkable, necessitating the rejection of the algorithm and the pursuit of an alternative solution. The verification process is inherently time-consuming due to the necessity of evaluating the compatibility of the conceptual solution with the actual objects present in the environment (Occurrences). The algorithm may require multiple iterations through potential solutions, discarding those that are infeasible, before identifying a viable option. This iterative process of generating and verifying solutions introduces an additional layer of complexity and time consumption to the algorithm.

It is imperative that these challenges be addressed in order to create an efficient algorithm.

5.Improved algorithm description

To address the challenges mentioned earlier, an improved version of the algorithm was developed to tackle more complex problems defined using the General Environment Description Language (GEDL). In the initial version of the algorithm, three critical issues were identified: resource inefficiency, the risk of infinite loops, and difficulties in real-life application. The improved version specifically focused on resolving the first two issues, which were crucial for enhancing the algorithm's performance and reliability.

The problem of resource inefficiency arose from the large number of operations and loops that the algorithm executed during the exploration process. As the algorithm evaluated possible actions and resulting states, the sheer volume of these operations led to significantly prolonged execution times, making the process less efficient and scalable for complex environments. Additionally, the potential for infinite loops became apparent when the algorithm repeatedly cycled through certain states without making meaningful progress toward the goal. For example, the algorithm might get stuck in a loop where it continuously opens and closes a door, failing to advance toward the final state. This issue not only wasted computational resources but also hindered the algorithm's ability to find a viable solution.

To overcome these challenges, several key enhancements were proposed and implemented. One of the most significant improvements was the introduction of a state queue. This queue acts as a central mechanism for managing the states encountered during the simulation. The algorithm now begins by taking the first state from the queue and appending all possible resultant states that can be achieved from this initial state. Before adding any new state to the queue, the algorithm checks whether this state has already been encountered in previous iterations. If the state has been encountered before, it is discarded to prevent the algorithm from entering an infinite loop. On the other hand, if the state is new, it is added to the queue, allowing the exploration process to continue. This approach effectively eliminates the risk of repetitive cycles and ensures that the algorithm makes continuous progress toward the final goal.

In addition to addressing infinite loops, further optimizations were implemented to reduce resource consumption. One significant optimization involved the use of references instead of duplicating instances, such as a washing machine, at each stage of the process. In the initial version, each state transition often involved creating cloned and slightly modified copies of the environment, which contributed to the resource inefficiency. By utilizing references, the algorithm can now directly address the initial state and the set of modifications made to reach the current state, rather than creating multiple copies. This approach substantially reduces the overhead associated with state transitions and minimizes the number of environmental states that need to be considered, thereby enhancing the overall efficiency of the algorithm.

With regard to the third issue, namely the inability to verify the applicability of a conceptual solution in real-world scenarios, the improved algorithm incorporates a

mechanism that systematically checks for the presence of required instances, features, and relationships in real-world occurrences subsequent to the generation of a conceptual solution.

The process commences with the algorithm formulating a step-by-step solution based on the concepts delineated within the GEDL framework. Once a conceptual solution has been established, the algorithm proceeds to verify its feasibility in a real-world environment. In particular, the solution is cross-referenced with the available instances, thus ensuring that the requisite features and relationships are present in the occurrences.

In the event that the required instances or relationships are absent, the algorithm identifies and presents a list of these missing assets, indicating that the proposed solution is not viable in the current environment. This feature enables a more informed evaluation of the solution's practicality, identifying the gaps that must be addressed for successful implementation. Conversely, if all the requisite instances and relationships are present, the step-by-step solution is deemed potentially executable, indicating a higher probability of success in a real-world context.

This enhancement not only enhances the algorithm's ability to solve problems but also provides a diagnostic tool that can guide users in adapting or refining their solutions based on the actual resources and constraints of the environment. The updated version of the algorithm is presented in Figure 3.

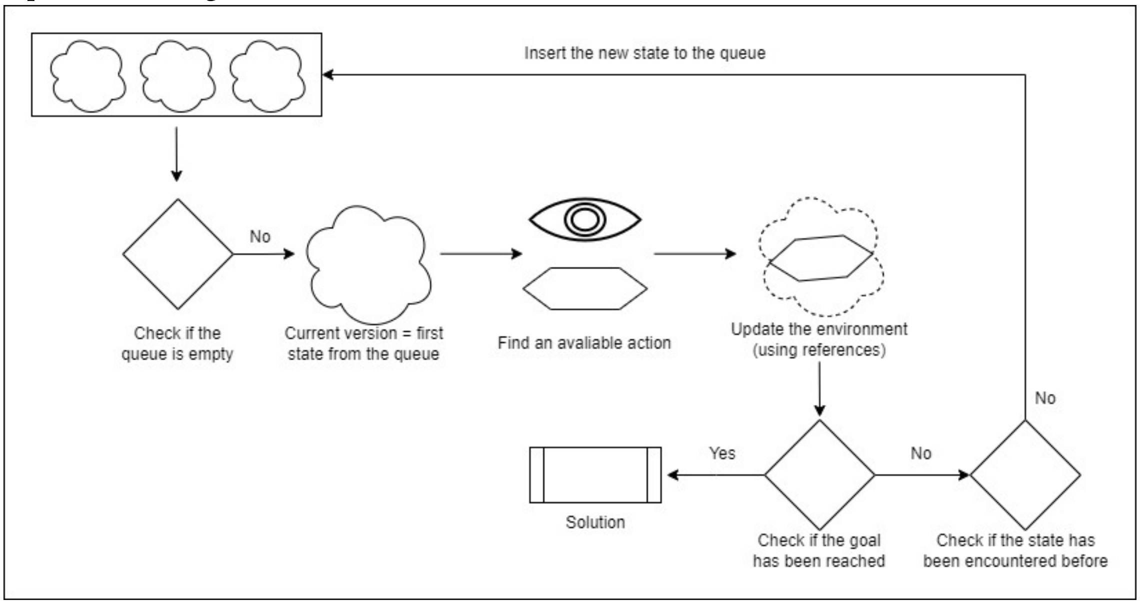
Insert the new state to the queue
No
Check if the queue is empty
Current version = first state from the queue
Find an avaliable action
Update the environment (using references)
No
Yes
No
Solution
Check if the goal has been reached
Check if the state has been encountered before

Figure 3: Updated process of finding a solution

6. Results

The enhanced algorithm was evaluated in a scenario that involved a washing machine, laundry, a basket, and a manipulator. This scenario was designed to assess the algorithm's capacity to effectively address complex problems within dynamic environments. This example demonstrated the significant advancements achieved by the algorithm.

Firstly, the implementation of the state queue effectively resolved the issue of infinite loops, such as those caused by repeatedly opening and closing doors or drawers. In earlier versions of the algorithm, these actions could result in an infinite loop, necessitating the introduction of artificial constraints to prevent such behaviour. However, the introduction of the queue has enabled the algorithm to rapidly identify and discard repetitive states. The initial iteration of any loop is promptly identified, thus ensuring that the algorithm does not expend resources on redundant actions. This improvement has not only maintained the algorithm's universality but has also considerably reduced the execution time.

Furthermore, the intelligent administration of components, including instances, features, and relationships, has resulted in notable enhancements in processing speed. The algorithm is now able to propose a conceptual solution with greater rapidity than was previously possible, due to its more efficient handling of these elements. In the washing machine scenario, the initial conceptual solution proposed by the algorithm was accurate and executable in real-world contexts, thereby demonstrating the efficacy of the enhancements.

Furthermore, the algorithm's flexibility was demonstrated through a test in which the basket containing laundry was replaced with an empty one. The queue mechanism promptly identified that the required actions could not be executed, resulting in the prompt depletion of the queue. This resulted in the identification of missing elements, specifically the laundry and the relationship "contains" (as in "basket contains laundry"). The algorithm's capacity to detect and report such deficiencies highlights its enhanced utility in navigating diverse environmental setups.

In conclusion, the modifications introduced have enhanced the algorithm's adaptability and efficiency, allowing it to operate in a broader range of environments while maintaining robust performance in complex, dynamic scenarios.

7. Direction of future work

In subsequent research, further challenges will be investigated with a view to further refining this approach. A crucial consideration is the manner in which the algorithm should respond when a necessary instance within the conceptual system is absent in the real-world occurrences. This scenario gives rise to a number of important questions. The question thus arises as to whether the algorithm should attempt to solve the problem once more, this time without relying on concepts that require the use of unavailable instances. Furthermore, if it is feasible to obtain or create the missing instance, should additional steps be incorporated into the solution to account for its production? The resolution of these questions will be crucial to the development of a more adaptable and intelligent problem-solving system.

To enhance the algorithm's efficiency, future research will also focus on refining the search space through the utilization of Individual Knowledge Fragments (IKFs). By specifying the instances available in the environment, IKFs can assist in narrowing down the search, thereby reducing the computational burden by limiting the algorithm's focus to the relevant concepts and instances. This approach will guarantee that the algorithm exclusively contemplates solutions that are based on the actual resources that are currently available, thereby enhancing both the speed and the accuracy of the process.

A further significant challenge is posed by the algorithm's capacity to identify implicit parameters essential for problem-solving, particularly in instances where such parameters are not explicitly specified. To illustrate, if the task is to cut a hole in a wooden plank of a specific size to fit a given ball, a human being can deduce that the diameter of the hole should be the same as that of the sphere. However, this kind of logical inference is not a straightforward process for an algorithm. Future work will investigate methods of equipping the algorithm with the capability to derive such implicit parameters, potentially through the integration of domain-specific knowledge or heuristic reasoning techniques. This will facilitate the algorithm's ability to process a more expansive range of tasks where not all requisite details are explicitly defined, thereby enhancing its versatility and problemsolving capabilities.

Further work will be carried out to improve the algorithm's capacity to process missing instances or relationships when a conceptual solution is not viable in real-world scenarios. It is possible to enhance the presented approach, particularly in the context of identifying solutions within occurrences. In order to address this challenge, it is proposed that evolutionary algorithms be integrated into the problem-solving framework. Evolutionary algorithms, which are inspired by the process of natural selection, offer a powerful method for optimizing solutions in complex environments. In this context, the sequence of actions (steps) can be represented as a genome, with each gene corresponding to a specific action or decision. It is anticipated that the incorporation of evolutionary algorithms will result in a number of notable benefits, including the capacity to adapt to complex environments containing multiple instances and varying constraints. The algorithm would be better equipped to handle these environments by evolving strategies in an adaptive manner, with each strategy being tailored to the specific characteristics of the environment in question. Over successive generations, the evolutionary algorithm would progressively enhance its solutions, learning from past successes and failures. It seems plausible to suggest that the iterative refinement process is likely to result in the generation of more reliable solutions.

8.Summary

The objective of this work was to develop and refine an algorithm capable of solving problems within environments described by the General Environment Description Language (GEDL). The initial version of the algorithm was effective at the conceptual level, generating solutions through the exploration of potential actions and state transitions. However, several significant obstacles were identified that restricted its applicability in real-world scenarios. These challenges included inefficiency in the use of resources, the risk of infinite loops, and difficulties in translating conceptual solutions into practical actions within real-world occurrences.

In order to surmount these limitations, a number of pivotal enhancements were introduced to the algorithm. Firstly, a state queue was implemented with the objective of providing a systematic means of managing and tracking the states encountered during the simulation. This mechanism proved an effective means of mitigating the issue of infinite loops, ensuring that previously encountered states were not revisited. Secondly, the consumption of resources was markedly decreased through the utilization of references in contrast to the replication of instances of objects at each stage of the problem-solving process. This optimization resulted in a notable enhancement in the efficiency and scalability of the algorithm. In addition, the algorithm was provided with the capacity to identify the absent elements, thereby reinforcing the necessity of evaluating the viability of conceptual solutions in authentic contexts.

Despite these advancements, the algorithm still faces challenges in applying conceptual solutions to complex real-world environments where multiple instances and varying constraints must be considered. To address this issue, future work will concentrate on integrating evolutionary algorithms into the problem-solving framework. Evolutionary algorithms, which are inspired by the principles of natural selection, represent a promising approach to optimising solutions within complex environments.

Furthermore, additional challenges will be investigated through further research. One area of focus will be to determine whether it would be preferable to re-solve the problem or to incorporate additional steps to create or acquire the necessary instance. A further crucial element will be narrowing down of the environment to the pertinent concepts and instances, thereby enhancing the algorithm's efficiency. Ultimately, efforts will be made to augment the algorithm's capacity to infer implicit parameters essential for problem-solving, particularly in instances where such parameters are not explicitly provided.

In conclusion, the proposed improvements and the planned future directions are designed to develop a more robust and versatile algorithm capable of solving complex problems across a variety of environments. By establishing a connection between conceptual planning and real-world execution, this work contributes to the advancement of automated problem-solving systems that are more generalisable and effective.

References

[1] A.K. Tyagi, T.F. Fernandez, S. Mishra, S. Kumari, (2021). Intelligent Automation Systems at the Core of Industry 4.0. In: Abraham, A., Piuri, V., Gandhi, N., Siarry, P., Kaklauskas, A., Madureira, A. (eds) Intelligent Systems Design and Applications. ISDA 2020. Advances in Intelligent Systems and Computing, vol 1351. Springer, Cham. doi: 10.1007/978-3-030-71187-0_1

[2] N. Bąkowska, K. Zatwarnicki, Solving problems using the General Environment Description Language, in: Proceedings ITTAP'2023: 3rd International Workshop on Information Technologies: Theoretical and Applied Problems, November 22–24, 2023, Ternopil, Ukraine, Opole, Poland

[3] K. Zatwarnicki, W. Pokuta, A. Bryniarska, A. Zatwarnicka, A. Metelski, E. Piotrowska, General Environment Description Language, Applied Sciences. January 2021; 11(2):740. Dou: 10.3390/app11020740

[4] B. Goertzel, I. Matthew, J. Wigmore, The Architecture of Human-Like General Intelligence, In: Theoretical Foundations of Artificial General Intelligence, Atlantis Thinking Machines, vol 4. Atlantis Press, Paris, doi: 10.2991/978-94-91216-62-6_8.

[5] M. Thielscher, (2010). A General Game Description Language for Incomplete Information Games.. in: Proceedings of the AAAI Conference on Artificial Intelligence, doi: 10.1609/aaai.v24i1.7647

[6] Thompson, Tommy & Ebner, Marc & Schaul, Tom & Levine, John & Lucas, Simon & Togelius, Julian. (2013). Towards a Video Game Description Language. Dagstuhl Follow-ups. 6. 85. 10.4230/DFU.Vol6.12191.i.

[7] M. Thielscher, (2017). GDL-III: A Description Language for Epistemic General Game Playing. International Joint Conference on Artificial Intelligence, doi: 10.24963/ijcai.2017/177

[8] T. Konnerth, B. Hirsch, S. Albayrak, JADL – An Agent Description Language for Smart Agents. In: Baldoni, M., Endriss, U. (eds) Declarative Agent Languages and Technologies IV. DALT 2006. Lecture Notes in Computer Science, vol 4327. Springer, Berlin, Heidelberg. doi: 10.1007/11961536_10.

[9] F. Bergenti, E. Iotti, S. Monica A. Poggi, Agent-Oriented Model-Driven Development for JADE with the JADEL Programming Language. Computer Languages, Systems & Structures, June 2017, doi: 50. 10.1016/j.cl.2017.06.001.

[10] H. Lin, Z. Wang, J. Ma, Y. Liang (2023) MCU: A Task-centric Framework for Open-ended Agent Evaluation in Minecraft, Workshop on Agent Learning in Open-Endedness (ALOE) at NeurIPS 2023, arXiv:2310.08367v1

[11] Z. Wang, S. Cai, A. Liu, Y. Jin, J. Hou, B. Zhang, H. Lin, Z. He, Z. Zheng, Y. Yang, X. Ma, Y. Liang (2023), JARVIS-1: Open-world Multi-task Agents with Memory-Augmented Multimodal Language Models, arXiv preprint arXiv:2311.05997

[12] OpenAI. Available online: https://openai.com/ (accessed on 20 August 2024)