Adaptive General Search Framework for Games and Beyond

Chiara F. Sironi
Game AI and Search group
Department of Data Science and Knowledge Engineering
Maastricht University
Maastricht, The Netherlands
c.sironi@maastrichtuniversity.nl

Mark H. M. Winands
Game AI and Search group
Department of Data Science and Knowledge Engineering
Maastricht University
Maastricht, The Netherlands
m.winands@maastrichtuniversity.nl

***Abstract*—The research field of Artificial General Intelligence (AGI) is concerned with the creation of adaptive programs that can autonomously address tasks of a different nature. Search and planning have been identified as core capabilities of AGI, and have been successful in many scenarios that require sequential decision-making. However, many search algorithms are developed for specific problems and exploit domain-specific knowledge, which makes them not applicable to perform different tasks autonomously. Although some domain-independent search algorithms have been proposed, a programmer still has to make decisions on their design, setup and enhancements. Thus, the performance is limited by the programmer's decisions, which are usually biased. This paper proposes to develop a framework that, in line with the goals of AGI, autonomously addresses a wide variety of search tasks, adapting automatically to each new, unknown task. To achieve this, we propose to encode search algorithms in a formal language and combine algorithm portfolios with automatic algorithm generation. In addition, we see games as the ideal test bed for the framework, because they can model a wide variety of complex problems. Finally, we believe that this research will have an impact not only on the AGI research field, but also on the game industry and on real-world problems.**

***Index Terms*—adaptive search, search algorithm generation, Artificial General Intelligence, General Game Playing**

I. Introduction

Research in Artificial Intelligence (AI) has largely focused on the so-called "narrow AI", which consists in developing specialized programs that show intelligence only on specific tasks and in specific contexts. Some of these programs are quite successful, but in general, they incur the risk of overfitting to the problem. Often, they tend to be refined using domain-specific knowledge and engineering tricks, and usually most of the knowledge is coded by the programmer instead of being learned by the program. This leaves less room for actual AI and makes the programs not applicable to any other task, unless major modifications are performed first. This picture recently led researchers to concentrate on more general AI, defining a new research area called Artificial General Intelligence (AGI) [1], [2]. This research area tackles the creation of programs that are able to perform different complex tasks in different environments and learn how to adapt to perform each task.

Search and planning have been identified among the core competences that an AGI should possess [2]. Moreover, we are currently facing the need to plan ahead in increasingly more complex situations, and search has shown to be successful in many applications that involve sequential decision making, such as logistics [3], health care [4], [5], and structural engineering [6]. This motivates the interest in investigating search algorithms form an AGI perspective.

Numerous successful search algorithms and variants thereof have been developed so far. The minimax algorithm [7], for example, is the foundation of most of the tree-search algorithms that enabled computer programs to reach a good performance in many board games. From minimax, the popular $\alpha\beta$-search [8], that uses a pruning technique to speed up the search, was developed. Moreover, to address problems with only one agent that can make decisions, the popular A* search algorithm [9] and its variants have been developed. One of the successful applications of these algorithms is pathfinding [10].

However, in many circumstances search algorithms are designed with specific problems in mind. Often they take advantage of problem-specific heuristics, designed ad-hoc by a problem expert. In order to address a different task, such algorithms might require the intervention of the programmer or a different domain expert. More general search approaches have started developing after the use of Monte-Carlo evaluations was proposed [11]. Notably, one of the most successful ones has been the Monte-Carlo tree search (MCTS) algorithm [12], which uses Monte-Carlo simulations instead of domain-specific heuristics to evaluate states. Research on this algorithm has particularly flourished also thanks to the field of General Game Playing (GGP) [13], which aims at creating programs that can address a wide variety of games without knowing them in advance. In this context, search algorithms that do not necessarily need problem-specific knowledge, like MCTS, have been investigated.

However, for this kind of algorithms, research has shown that there is no single setting or combination of enhancements working best on every possible game [14], [15]. Moreover, although aiming to reduce human intervention, for most of

978-1-6654-3886-5/21/$31.00

the approaches proposed so far for GGP, the programmer is still required to make decisions on the design of the algorithm. For example, s(he) has to decide in advance which algorithm to use, which enhancements to activate and which parameter settings to use.

To overcome these limitations, some research on adaptive agents has been performed. For example, Swiechowski *et al.* [16] proposed an online mechanism for GGP that learns which, among a portfolio of action selection strategies for MCTS, is the most suited strategy for the current game. The work of Lucas *et al.* [17] investigates the use of evolutionary algorithms to adapt online the weights that bias action selection during the playout phase of MCTS, applying the resulting algorithm to general video game playing (GVGP). Another example is the work of Mendes *et al.* [18], who proposed the use of algorithm portfolios for GVGP. Instead of having an expert chose an algorithm in advance, a meta-agent trained offline selects among a portfolio of possible search algorithms the one that is predicted to perform best on the given game. Finally, Sironi *et al.* [19] proposed an approach to automatically tune MCTS parameters online for each new game, instead of letting the programmer tune them manually offline.

Despite automatizing some of the decisions, these approaches still require the intervention of the programmer to choose, for example, which MCTS strategies to consider, which parameters to tune, which domains to use for the values, or how to populate the portfolio. This implies that the final performance of the algorithms is still limited by human decisions, which are usually biased. In fact, programmers tend to make decisions depending on the problems they anticipate the search will tackle, but new and unexpected problems might arise in the future. This paper introduces the vision behind the project that deals with this gap. The main idea is to develop a framework that, in line with the goals of AGI, can autonomously address a wide variety of search tasks (including, but not limited to games), adapting automatically to each new, unknown task.

Initial research on agents that automatically adapt to each new game, like the one discussed above, has shown how research in the direction of automatic adaptation, promoted also by the field of AGI, is becoming more and more relevant also for GGP. Self-adaptive agents were also identified as an important research challenge during the 2019 Dagstuhl seminar on Artificial and Computational Intelligence in Games [20]. Moreover, the need to look for more advanced artificial game players has also been highlighted by Gaina *et al.* [21], who envisioned the development of an agent that goes beyond classical GGP agents. They propose to develop an "always on" game-playing agent, that learns from experience, continuously getting better and interacting with the world. The framework proposed in this paper aims to extend the state-of-the-art in this direction.

The rest of the paper is structured as follows. First, Section II gives a formal definition of the search problems that the proposed framework will address. Next, an overview of the structure and components of the framework is given in Section III. Section IV discussed the challenges that developing such framework brings about, mentioning possible methods that can be used to address them. Subsequently, Section V argues that games are the ideal test bed for the framework, and Section VI discusses the relevance and potential impact of the proposed research on AGI, the game industry and other real-world applications. Finally, Section VII concludes the paper.

II. Search Problem Definition

The search problems that this research is aiming to address can be defined by the following components:

- A set of m actors $I = \{1, ..., m\}$, that can make decisions in the search problem.
- A set of all the possible states of the problem S.
- For each *actor* $i \in I$, a set with all the actions that the actor can perform in any state of the problem, A_i.
- An *initial state* s_0.
- An *actions* function that given a state $s \in S$ and an actor $i \in I$ returns all the *actions* that i can perform in s: $Actions(s, i) : S \times I \rightarrow \mathcal{P}(A_i)$, where $\mathcal{P}(A_i)$ indicates the power set of A_i.
- A *successor* function that given a state s and an action for each actor, $\vec{a} = \langle a_1, ..., a_m \rangle$, returns one of the possible next states according to the probability distribution over all the possible next states: $Successor(s, \vec{a}) : S \times Actions(s, 1) \times ... \times Actions(s, m) \rightarrow S$. For deterministic problems this function will always return the same state.
- A *goal* function that determines if state s is a goal (thus, the search is over): $Goal(s) : S \rightarrow \{true, false\}$.
- A function that given a state s returns the payoff obtained by each actor in such state: $Payoff(s) : S \rightarrow \mathbb{R}^m$.

The framework proposed in this paper will be able to address any search problem for which the above components are defined. Note that it is not essential for the framework to have access to the complete set of states S and the sets of available actions for each player, $\{A_1, ..., A_n\}$, as long as it has access to the *actions* and *successor* functions. This means that the framework must have access to a forward model of the problem. However, in the future it could be extended with a mechanism that learns the forward model [22], so that any approach used by the framework can still be applicable.

The above definition also implies that the problems are addressed with discrete time steps, and that each actor has to make a decision in each time step. However, this definition enables us to model problems where not all actors perform an action at each time step, by introducing a *nil* action that has no effect on the state. This is the same approach that is often used to define games in GGP. We also assume that both the state and action spaces are discrete. However, the framework is applicable to problems with continuous aspect after appropriate discretization.

III. Overview of the Framework

This paper proposes to create a framework that can autonomously address a wide variety of search tasks, adapting

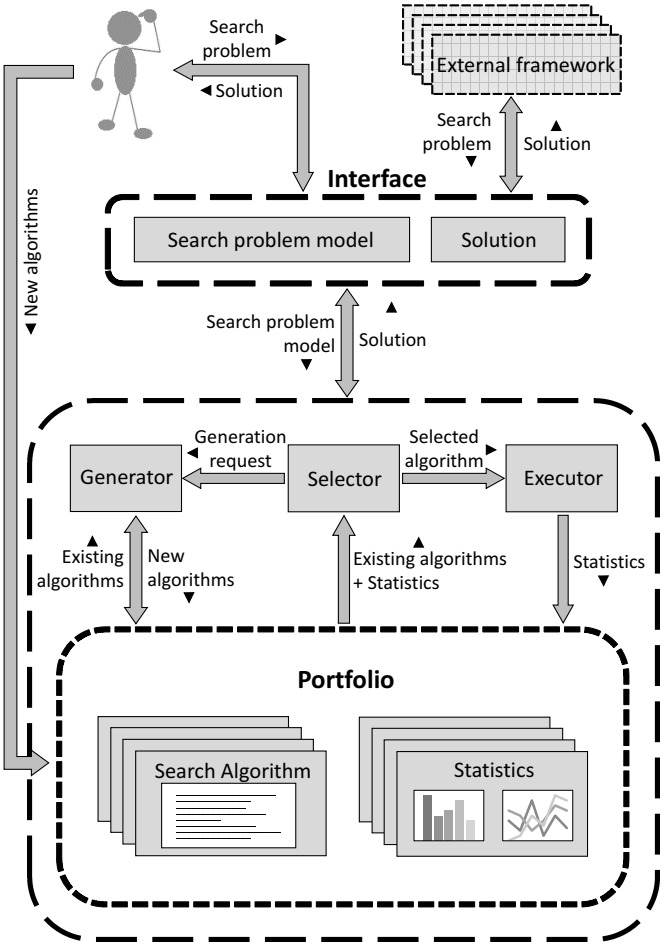

Fig. 1. Overview of the adaptive general search framework

its approach to each new task automatically. The main innovation is that such framework is based on the combination of algorithm portfolios [23] with automatic algorithm selection [24] and automatic algorithm generation [25]. Research into combining algorithm portfolios with automatic algorithm generation is now possible thanks to previous work in both areas, which we can combine and extended towards domain-independent approaches.

Figure 1 shows the overview of the proposed framework. Because there is no single search algorithm that performs best on every search task [23], the framework selects for each task a good enough search algorithm from a portfolio and generates new ones when existing ones do not give good results.

The framework includes a portfolio of algorithms for which statistics on past performance are memorized. To make the framework independent of specific programming languages, algorithms in the portfolio are coded in a predefined formal language. When a new search problem is presented, the Selector looks in the portfolio to find a suitable algorithm depending on the available statistics. If it deems no algorithm to be promising for the task, the Selector will query the Generator to create new algorithms for the portfolio. The Generator might also use existing algorithms as starting point for the generation. This interaction might repeat multiple times, until the Selector finds a good enough algorithm for the problem or until a given time budget expires. At this point, the most suitable algorithm found so far is given to the Executor, which will run it, return a solution, and memorize in the Portfolio the statistics about the algorithm's performance. To be in line with the goals of AGI, the framework should be easy to use with external frameworks that encode search problems of different nature (e.g. GGP frameworks). For this reason, an interface is offered, that defines the format for the model and the solution of a search problem that external frameworks should adhere to, in order to be usable with the system. This model includes the components of the problem mentioned in Section II. Finally, a user can interact with the framework in two ways. (i) S(he) can provide a problem specification and can query the framework for a solution, adhering to the given interface. (ii) S(he) can contribute to the portfolio by encoding new search algorithms. Note that the last interaction is optional, as the framework is able to generate new algorithms autonomously. The human can enrich the knowledge of the system, but it is not anymore essential for its design.

IV. Challenges

From the description of the system, we can identify the following three main research challenges:

- Developing a formal language to encode search algorithms.
- Developing methods that decide how to select good enough search algorithms to address each given task without using any domain-specific information.
- Developing methods that automatically generate new search algorithms.

Subsections IV-A, IV-B and IV-C discuss each of the tree aspects, respectively, giving an indication of feasible methods that can be used to address them.

A. Algorithm Representation

The framework requires a language that encapsulates all the concepts that are necessary to define search algorithms, and that restricts the search space to such algorithms only (or to a reasonably large subset of such algorithms). The following properties are desirable for this language:

- *Simplicity:* algorithm descriptions should be easy to create and modify. Thus, they should not require too many tokens to be encoded.
- *Clarity:* algorithm descriptions should be easy to understand by humans, to prevent generated algorithms from becoming black boxes. This property is particularly relevant considering the interest in explainable AI, and, more recently, in explainable search [26].
- *Generality:* the language should be general enough to enable the encoding of as many search algorithms as possible.
- *Extensibility:* it should be easy to extend the language to include more functionality, so that the set of search algorithm that can be represented can increase.

- *Evolvability:* the language should facilitate the creation of new algorithms by evolving existing ones. It should ensure a high probability that the generated algorithms are reasonable.
- *Efficiency:* the language should facilitate the efficient execution of the encoded algorithms.

Defining the language using a class grammar [27] could satisfy these requirements. A class grammar is a formal grammar that maps to the class hierarchy of the underlying code, offering the same functionality, but abstracting the implementation details. This means that any object-oriented programming language can be used to implement the underlying code. This approach would also make the language easily extensible to include algorithms beyond search. A class grammar has also been successfully used to define game descriptions for the Ludii General Game Playing framework [28], and for games it has been shown to provide the same properties mentioned above.

To give an idea of how the description of an algorithm would look like using a class grammar, the following is a possible partial description of an instance of the MCTS algorithm:[1]

```
(SearchAlgorithm "MCTS"
  (ActionSelection "UCT"
    (Condition inTree)
    (Select max)
    (ValueFunction
            UCB1(s,a) = Q̄(s,a) + C * sqrt(ln(n(s))/n(s,a)))
  )
  (ActionSelection "random"
    (Condition outTree)
    (Select max)
    (ValueFunction RND(s,a) = random(0,1))
  )
          ⋮
  (ActionSelection "maxAvg"
    (Condition final)
    (Select max)
    (ValueFunction Q̄(s,a))
  )
)
```

Different action selection strategies are defined. The first is the one used during the selection phase of MCTS, which is used in the tree built so far by the algorithm and uses the popular UCB1 value function [29]. The second is the one used during the playout phase of MCTS. The third is the one used at the end of the search to choose the final action to perform. For each action selection strategy the description specifies a condition that has to be satisfied to use such strategy, a function that computes the value of each action in the current state, and the criterion used to select an action based on the value function. In this case, all action selection strategies are choosing the action for which the value function returns the maximum value. Also note that formulas may have parameters, such as the exploration constant C in the UCB1 formula. The framework will also be able to set the values of such parameters, creating multiple instances of the same algorithm.

[1]This description is meant to illustrate how we envision the general structure of the language. A more extensive study to formally define the grammar behind the language is object of future research.

B. Algorithm Selection

Algorithm portfolios have been extensively used to address scenarios in which there is no single algorithm performing optimally on all the given problems [23]. Portfolios have been applied to many combinatorial search problems [24]. Notably, they have been successful at efficiently finding solutions for instances of the SAT problem [30]. Particularly in GGP, portfolio approaches and automatic algorithm selection have been applied to general video game playing, showing the benefits of relying on a combination of different algorithms [18], [31]–[33].

Notably, choosing the best algorithm for a given task is one of the main challenges of portfolio approaches. Usually, portfolios are composed manually, or by automatically selecting in a restricted algorithm space. The framework we are envisioning aims to go beyond this limitation, by letting the program search in the space of all reasonable search algorithms to generate new ones. In addition, algorithm selection mechanisms usually exploit domain-specific features to determine which algorithm is most suited for a problem, while our approach will look for general features that are applicable independently of the problem's domain.

One of the most common solutions to the algorithm selection problem consists of the following three steps [34]:

1) Find task features that are representative of algorithm performance.
2) Evaluate the available algorithms on a subset of tasks with different features.
3) generate a model that predicts algorithm performance as a function of the features.

A similar approach will be investigated in the context of the proposed framework, with the main difference being the need to define and extract general features, thus domain-independent. These features could be defined by looking at the search problem structure and characteristics, rather than by looking at the problem domain. For example, by having access to the forward model of the problem, the framework could extract information about the branching factor of the search space, the number of agents or the number of available actions. It could also try to analyze the payoff landscape.

Another difference with the approach proposed by Leyton-Brown *et al.* [34] is that our framework will repeatedly perform steps (2) and (3), improving over time the quality of the algorithms in the portfolio and increasing the amount of statistics collected for each algorithm. This introduces other challenges for developing such mechanism. First, we have to find a suitable mechanism to decide which algorithms to keep in the portfolio. The more algorithms are in the portfolio, the

higher the chance that one of them is suitable for a given task. However, at the same time, it becomes more time consuming to compute a performance prediction model. Second, we have to decide which statistics to memorize, because there are multiple aspects that define algorithm performance. For example, we could look at the payoff obtained by each algorithm, or at the running time.

C. Algorithm Generation

Clune [25] argues that the creation of AI-generating algorithms should be considered as a new, independent scientific grand challenge, and might be the key to finding true AGI. Research on automatic generation of search algorithms that can perform many heterogeneous tasks without human intervention contributes to addressing this new grand challenge.

Automatic generation has been investigated to different extents and with different levels of generality for data processing algorithms [35], reinforcement learning [36], [37], and distributed algorithms for multi-agent systems [38]. However, not much work has been performed on automatic generation of search algorithms, despite the wide variety of real-world applications that might benefit from it [3], [4], [6]. Automatic generation has been explored also at a higher level. For example, to synthesize generic programs by searching in the space of all programs that can be coded in a given programming language [39], and also to generate programs by induction [40] (i.e. using given input-output combinations to automatically discover programs that are capable to generalize to previously unseen inputs). The approach proposed in this paper, although similar to program synthesis, initially focuses on search algorithms. This restriction enables us to model more complex behaviors [39] and generate advanced search algorithms. Moreover, the proposed system uses a high-level algorithm representation language that is independent from any programming language. After having generated a new algorithm in this language, it can be mapped to any object-oriented language.

In this paper, we propose to create a method that autonomously searches in the space of search algorithms in order to generate new ones. This idea originated during a Dagstuhl seminar on Artificial and Computational Intelligence in Games [41]. Research on algorithm generation is rather limited. However, approaches that seem reasonable to tackle this problem come from the field of Evolutionary Computation. Evolutionary algorithms are powerful approaches for problems with large search spaces, given their exploratory nature [42]. This makes them promising to be applied to the large space of possible search algorithms. A search algorithm could be considered as an individual and all the concepts expressed in the algorithm description language presented in Section IV-A could be considered as genes. Moreover, the hierarchical structure of the language enables us to identify genes at different levels, and evolve algorithms with different levels of granularity. Looking at the description of the MCTS algorithm given in Subsection IV-A, we could consider to change an entire strategy, or change only some of its components. Moreover, it would be possible to also evolve functions, such as the ones that evaluate actions. For instance, by following an approach similar to the one proposed by Bravi *et al.* [43] to evolve variants of the UCB1 formula.

Deep learning approaches could also be investigated to generate search algorithm, due to their ability to generate new reasonable artifacts, after being trained on a set of examples [44]. However, they might be more suitable at a later state of the project, because they require a massive volume of data to train on. At the start of the project, we might not have a sufficient variety of available search algorithm to give as example, even if we would manually encode existing search algorithms.

One final challenge to consider when generating new search algorithms is how to evaluate their fitness. It is not trivial to evaluate an algorithm. As we discussed for algorithm selection, we could look at the win rate of the algorithm or at its running time. However, for newly generated algorithms past performance statistics will not be available. Therefore, a different fitness evaluation is necessary. A possibility could be to evaluate the generated algorithm on the search problem that is being considered, while trying to solve it. Thus, not looking at the overall performance but at the quality (e.g. payoff) of the states it visits. Additionally, statistics of existing algorithms could be used to estimate how certain algorithm components might perform. For example, if the new algorithm is using an action selection strategy that is part of other algorithms in the portfolio, statistics could be combined to estimate the performance of such strategy.

V. Games as Test Environments

Games can model a wide variety of computationally hard problems, while still providing a controlled and well-formed environment for testing [45], [46]. We can consider games as a reasonable subset of all the search problems that we would like our framework to address. Therefore, we see them as a suitable test environment for this project.

Moreover, due to the renewed interest in GGP research in the past 15 years, we now have an abundance of search problems (games and video games) that can be used to test such approaches. Numerous GGP frameworks are openly available, providing large sets of games of different nature. For example, the Stanford GGP framework [13], [47] provides access to numerous abstract games, by generating their forward model from their rules. Another example is the Ludii framework [28], [48], which, to date, offers around 700 abstract and historical games, and is continuously being extended with more games. Real-time arcade-style video games with non-determinism are offered by the General Video Game AI (GVGAI) framework [49], [50]. Finally, research in GGP is continuously expanding and new frameworks are being developed, extending to other types of games. For instance, the Stratega framework [51], [52] has been recently proposed as a GGP framework for real-time strategy games.

VI. Relevance and Impact of the Research

The proposed research has multiple stakeholders. It has the potential to have impact on the AGI research field (Subsection VI-A, on the game industry (Subsection VI-B, and in general on many real-world problems (Subsection VI-C).

A. Artificial General Intelligence

For many years, research in Artificial Intelligence (AI) has largely focused on "narrow AI". However, the interest is now extending to AI programs that can autonomously adapt to perform many heterogeneous tasks, and the field of AGI is attracting more and more attention. From a scientific perspective, the outcomes of this project will contribute to expand the state-of-the-art of AGI approaches. Although used to represent, select and generate search algorithms, the methods that will be developed as part of this project will be domain-independent. This means that they can have an impact on other research areas, where they could be applied to develop adaptive systems based on different types of algorithms. For example, the same approach could be used to automatically generate other machine learning algorithms. Moreover, the framework that this proposal aims to develop has the potential to generate new search algorithms, which might enable researchers to solve problems for which no existing algorithm can find a good enough solution.

Finally, AGI aims to combine multiple capabilities, of which search and planning are only a subset. The proposed framework has the potential to be integrated in AGI frameworks that provide modules that implement other capabilities (e.g. communication, emotion, creation, reasoning, etc. [2]). Such a framework may enable machines to perform tasks that were not feasible in the past, by combining multiple competences.

B. Gaming Industry

The outcome of the project presented in this paper has the potential to create value for the gaming industry. Video games are getting more and more realistic and sophisticated, demanding smarter AI characters that are able to deal with increasingly complex environments. Moreover, if AI characters show unrealistic behaviors, the interest of the players in the game will decrease and the game will lose its entertainment value. Search algorithms can be used to model believable AI characters in commercial applications, and there have been already some successful examples of this. In the Spades mobile phone game by AI Factory MCTS has been used to create engaging AI opponents and allies [53], and has later been improved to emulate human play [54]. Moreover, MCTS has been used in the development of the AI system for the online turn-based strategy game Prismata, by Lunarch Studios [55], and in the development of the AI system of Total War: Rome II, a strategy game developed by Creative Assembly and published by Sega [56].

Potential applications of the proposed research to games are not only limited to the development of AI characters for the game industry. Search algorithms can also be useful during the process of game design and game content generation. A possibility is to use them evaluate generated games. For instance, Browne [57] computes game evaluation metrics using the performance of UCT-based search on such games. Other approaches apply MCTS to directly generate games or game content. For example, it has been used to generate initial tile placements for Pentominoes puzzles [58] and to generate Sokoban puzzles [59]. In addition, it has been applied to automatically generate narratives [60], which could be used in virtual environments or in video games to dynamically generate new plot lines.

The proposed framework could be used to adapt search algorithms to the particular needs of the game developers, possibly generating new algorithms that better suit their needs, both for controlling game characters and for generating game content. Different algorithms could be generated by the framework that implement different behaviors for the AI characters. Moreover, the framework will be able to also generate algorithms that learn from and adapt to different in-game situations. This means that the algorithms could be used to implement adaptive game characters.

C. Real-World Problems

AGI programs that are able to cope with tasks of different nature without needing human intervention are suitable to be applied in many real-world scenarios. They are potentially useful for environments in which the type of tasks that the machine has to face might change over time, but they cannot be predicted or even imagined by the programmers. Moreover, such programs might be successful in environments for which it is too time consuming or even physically impossible to re-program the machine for each new type of task. For example, for a robot on a space mission there might be scenarios that the programmers have not thought about, therefore the robot should be able to learn by itself how to cope with them and how to perform the necessary tasks to deal with unforeseen situations that require planning. Moreover, programmers are not able to physically access the robot to re-program it once it has reached space.

Search is a technique that can be applied successfully in many domains of social and economic interest. Many real-world domains involve planning, optimization and decision-making, and there are many examples of successful application of search algorithms in these domains. There are applications that range from healthcare [4], [5], [61] to space exploration [62], [63] and energy management [64]. Various applications of search algorithms can be found also in logistic [3] and robotics [65], [66]. Many more examples can be mentioned, including structural engineering [6], protecting natural resources from illegal extraction [67], forecasting financial volatility of assets [68], managing wildfires [69], improving computer-aided retrosynthesis [70] and maximizing the performance of job scheduling heuristics [71]. Many more applications are being explored every day. A framework that adapts search algorithms to every new task following the direction of AGI has the potential to be applied to all the above mentioned fields, without requiring any reprogramming or human intervention.

VII. Conclusion

This paper presented our vision for a framework that can address a variety of search tasks by automatically adapting to perform each new task that is presented to it. Following the trend of AGI, we aim to extend the state-of-the-art search algorithms towards more general approaches that do not depend on domain-specific characteristics and do not require human intervention to adapt their behavior.

To implement the framework, we proposed to combine algorithm portfolios with automatic algorithm generation. This system can select appropriate algorithms for each task and can adapt them, and even generate new ones, when the existing ones are not performing well on the given task. To encode algorithms we proposed to use a high level language that defines the space where to search for new algorithms. The research proposed in this paper presents many challenges and we have discussed some methods that might be applied to address them. Moreover, we have identified games, and GGP frameworks in particular, as the ideal test bed for our research. Finally, we argued that the outcomes of the proposed research will benefit the AGI research field and the game industry, and be applicable also to many real-world problems.

The research proposed in this paper is meant to support and give new tools for the creation of general programs that present traits of AGI. Initially, we limit the scope of the framework to search algorithms, and its application to search problems for which a forward model is provided. However, we envision the outcomes of this research to provide approaches and mechanism that can be easily extended to include more types of algorithms and types of problems. The algorithm description language that will be developed will be easily extensible to include constructs for other types of algorithms. The algorithm selection and generation mechanisms that we envision, are domain independent, thus should be applicable to other types of algorithms without the need for major modifications. Finally, the framework could be extended to include more components, such as a component that is capable of learning the forward model of the search problem.

Acknowledgment

The authors would like to thank all the researchers and colleagues who helped shaping the research proposed in this paper and those who agreed to support the project. In particular, we thank the members of the Game AI & Search research group and of the Dynamic Game Theory research group of the Department of Data Science and Knowledge Engineering of Maastricht University for their valuable feedback.

References

[1] B. Goertzel and C. Pennachin, *Artificial General Intelligence*. Springer, 2007.

[2] B. Goertzel, "Artificial general intelligence: concept, state of the art, and future prospects," *Journal of Artificial General Intelligence*, vol. 5, no. 1, pp. 1–46, 2014.

[3] S. Edelkamp, M. Gath, C. Greulich, M. Humann, O. Herzog, and M. Lawo, "Monte-Carlo tree search for logistics," in *Commercial Transport*. Springer International Publishing, 2016, pp. 427–440.

[4] C. Laschet, J. op den Buijs, M. H. M. Winands, and S. Pauws, "Service selection using predictive models and Monte-Carlo tree search," *arXiv preprint arXiv:2002.04852*, 2020.

[5] J. Van Eyck, J. Ramon, F. Guiza, G. Meyfroidt, M. Bruynooghe, and G. Van den Berghe, "Guided Monte Carlo tree search for planning in learned environments," in *Asian Conference on Machine Learning (ACML)*, ser. JMLR Workshop and Conference Proceedings, C. Ong and T. Ho, Eds., vol. 29, 2013, pp. 33–47.

[6] L. Rossi, M. H. M. Winands, and C. Butenweg, "Monte Carlo tree search as an intelligent search tool in structural design problems," *Engineering with Computers*, pp. 1–18, 2021.

[7] J. von Neumann and O. Morgenstern, *Theory of Games and Economic Behavior*. Princeton, NJ, USA: Princeton University Press, 1944.

[8] D. E. Knuth and R. W. Moore, "An analysis of alpha-beta pruning," *Artificial Intelligence*, vol. 6, no. 4, pp. 293–326, 1975.

[9] P. E. Hart, N. J. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *IEEE Transactions on Systems Science and Cybernetics*, vol. 4, no. 2, pp. 100–107, 1968.

[10] N. Sturtevant and M. Buro, "Partial pathfinding using map abstraction and refinement," in *Twentieth National Conference on Artificial Intelligence (AAAI 2005)*, M. M. Veloso and S. Kambhampati, Eds., vol. 5, 2005, pp. 1392–1397.

[11] B. Abramson, "Expected-outcome: A general model of static evaluation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 12, no. 2, pp. 182–193, 1990.

[12] C. Browne, E. Powley, D. Whitehouse, S. M. Lucas, P. I. Cowling, P. Rohlfshagen, S. Tavener, D. Perez, S. Samothrakis, and S. Colton, "A survey of Monte Carlo tree search methods," *IEEE Transactions on Computational Intelligence and AI in Games*, vol. 4, no. 1, pp. 1–43, 2012.

[13] M. Genesereth, N. Love, and B. Pell, "General game playing: Overview of the AAAI competition," *AI Magazine*, vol. 26, no. 2, pp. 62–72, 2005.

[14] H. Finnsson, "Simulation-based general game playing," Ph.D. dissertation, School of Computer Science, Reykjavik University, Reykjavik, Iceland, 2012.

[15] C. F. Sironi, "Monte-Carlo tree search for artificial general intelligence in games," Ph.D. dissertation, Maastricht University, Maastricht, The Netherlands, 2019.

[16] M. Świechowski and J. Mańdziuk, "Self-adaptation of playing strategies in general game playing," *IEEE Transactions on Computational Intelligence and AI in Games*, vol. 6, no. 4, pp. 367–381, 2013.

[17] S. M. Lucas, S. Samothrakis, and D. Perez, "Fast evolutionary adaptation for Monte Carlo tree search," in *European Conference on the Applications of Evolutionary Computation*. Springer, 2014, pp. 349–360.

[18] A. Mendes, J. Togelius, and A. Nealen, "Hyper-heuristic general video game playing," in *2016 IEEE Conference on Computational Intelligence and Games (CIG)*. IEEE, 2016, pp. 94–101.

[19] C. F. Sironi, J. Liu, and M. H. M. Winands, "Self-adaptive Monte-Carlo tree search in general game playing," *IEEE Transactions on Games*, vol. 12, no. 2, pp. 132–144, 2020.

[20] C. F. Sironi, D. Ashlock, C. Browne, T. Schaul, H. Wang, and M. H. M. Winands, "Self-adaptive agents for general game playing," *Dagstuhl Report - Artificial and Computational Intelligence in Games: Revolutions in Computational Game AI*, vol. 9, no. 12, pp. 100–101, December 2019.

[21] R. D. Gaina, S. M. Lucas, and D. Perez-Liebana, "Project Thyia: A forever gameplayer," in *2019 IEEE Conference on Games (CoG)*. IEEE, 2019, pp. 1–8.

[22] A. Dockhorn and S. Lucas, "Local forward model learning for GVGAI games," in *2020 IEEE Conference on Games (CoG)*. IEEE, 2020, pp. 716–723.

[23] D. Souravlias, K. E. Parsopoulos, I. S. Kotsireas, and P. M. Pardalos, *Algorithm Portfolios: Advances, Applications, and Challenges*. Springer International Publishing, 2021.

[24] L. Kotthoff, "Algorithm selection for combinatorial search problems: A survey," in *Data Mining and Constraint Programming*. Springer, 2016, pp. 149–190.

[25] J. Clune, "AI-GAs: AI-generating algorithms, an alternate paradigm for producing general artificial intelligence," *arXiv preprint arXiv:1905.10985*, 2019.

[26] H. Baier and M. Kaisers, "Explainable search," in *2020 IJCAI-PRICAI Workshop on Explainable Artificial Intelligence*, 2020.

[27] C. Browne, "A class grammar for general games," in *International Conference on Computers and Games*. Springer, 2016, pp. 167–182.

[28] E. Piette, D. J. N. J. Soemers, M. Stephenson, C. F. Sironi, M. H. M. Winands, and C. Browne, "Ludii - the ludemic general game system," in *ECAI 2020: 24th European Conference on Artificial Intelligence*, G. De Giacomo, A. Catala, B. Dilkina, M. Milano, S. Barro, A. Bugarín, and J. Lang, Eds., vol. 325, 2020, pp. 411–418.

[29] P. Auer, N. Cesa-Bianchi, and P. Fischer, "Finite-time analysis of the multiarmed bandit problem," *Machine Learning*, vol. 47, no. 2–3, pp. 235–256, 2002.

[30] L. Xu, F. Hutter, H. H. Hoos, and K. Leyton-Brown, "SATzilla: portfolio-based algorithm selection for SAT," *Journal of Artificial Intelligence Research*, vol. 32, pp. 565–606, 2008.

[31] P. Bontrager, A. Khalifa, A. Mendes, and J. Togelius, "Matching games and algorithms for general video game playing," in *Twelfth Artificial Intelligence and Interactive Digital Entertainment Conference (AIIDE)*, N. Sturtevant and B. Magerko, Eds. AAAI Press, 2016, pp. 122–128.

[32] D. Ashlock, D. Perez-Liebana, and A. Saunders, "General video game playing escapes the no free lunch theorem," in *2017 IEEE Conference on Computational Intelligence and Games (CIG)*. IEEE, 2017, pp. 17–24.

[33] D. Anderson, C. Guerrero-Romero, D. Perez-Liebana, P. Rodgers, and J. Levine, "Ensemble decision systems for general video game playing," in *2019 IEEE Conference on Games (COG)*. IEEE, 2019.

[34] K. Leyton-Brown, E. Nudelman, G. Andrew, J. McFadden, and Y. Shoham, "A portfolio approach to algorithm selection," in *Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence (IJCAI)*, vol. 3, 2003, pp. 1542–1543.

[35] M. Kuhner, D. Burgoon, P. Keller, S. Rust, J. Schelhorn, L. Sinnott, G. Stark, K. Taylor, and P. Whitney, "Automatic algorithm generation," November 2002, uS Patent App. 10/097,198.

[36] J. Veness, K. S. Ng, M. Hutter, W. Uther, and D. Silver, "A Monte-Carlo AIXI approximation," *Journal of Artificial Intelligence Research*, vol. 40, pp. 95–142, 2011.

[37] M. Hutter, *Universal Artificial Intelligence: Sequential Decisions Based on Algorithmic Probability*. Springer Science & Business Media, 2004.

[38] S. Van Berkel, D. Turi, A. Pruteanu, and S. Dulman, "Automatic discovery of algorithms for multi-agent systems," in *Proceedings of the 14th Annual Conference Companion on Genetic and Evolutionary Computation*, 2012, pp. 337–344.

[39] S. Gulwani, O. Polozov, and R. Singh, "Program synthesis," *Foundations and Trends in Programming Languages*, vol. 4, no. 1–2, pp. 1–119, 2017.

[40] K. Ellis, C. Wong, M. Nye, M. Sable-Meyer, L. Cary, L. Morales, L. Hewitt, A. Solar-Lezama, and J. B. Tenenbaum, "DreamCoder: Growing generalizable, interpretable knowledge with wake-sleep Bayesian program learning," *arXiv preprint arXiv:2006.08381*, 2020.

[41] D. Ashlock, T. Schaul, C. F. Sironi, and M. H. M. Winands, "Representation in evolutionary computation for games," *Dagstuhl Report - Artificial and Computational Intelligence in Games: Revolutions in Computational Game AI*, vol. 9, no. 12, pp. 75–79, December 2019.

[42] D. Ashlock, *Evolutionary Computation for Modeling and Optimization*. Springer Science & Business Media, 2006.

[43] I. Bravi, A. Khalifa, C. Holmgård, and J. Togelius, "Evolving game-specific UCB alternatives for general video game playing," in *Applications of Evolutionary Computation*, ser. LNCS, G. Squillero and K. Sim, Eds., vol. 10199. Springer, 2017, pp. 393–406.

[44] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning*. MIT press Cambridge, 2016.

[45] T. Schaul, J. Togelius, and J. Schmidhuber, "Measuring intelligence through games," *arXiv preprint arXiv:1109.1314*, 2011.

[46] G. N. Yannakakis and J. Togelius, *Artificial Intelligence and Games*. Springer, 2018.

[47] S. Schreiber and A. Landau, "The General Game Playing base package," https://github.com/ggp-org/ggp-base, 2016.

[48] C. Browne, E. Piette, M. Stephenson, W. Crist, and D. J. N. J. Soemers, "Ludii general game system," https://ludii.games, 2021.

[49] D. Perez-Liebana, "The GVG-AI competition framework," https://github.com/GAIGResearch/GVGAI, 2018.

[50] D. Perez-Liebana, J. Liu, A. Khalifa, R. D. Gaina, J. Togelius, and S. M. Lucas, "General video game AI: a multi-track framework for evaluating agents, games and content generation algorithms," *IEEE Transactions on Games*, vol. 11, no. 3, pp. 195–214, 2019.

[51] A. Dockhorn, J. Hurtado-Grueso, D. Jeurissen, and D. Perez-Liebana, "Stratega - a general strategy games framework," in *AIIDE-20 Workshop on Artificial Intelligence for Strategy Games*, 2020.

[52] GAIGResearch, "Stratega," https://github.com/GAIGResearch/Stratega, 2021.

[53] D. Whitehouse, P. I. Cowling, E. J. Powley, and J. Rollason, "Integrating Monte Carlo tree search with knowledge-based methods to create engaging play in a commercial mobile game," in *Ninth Artificial Intelligence and Interactive Digital Entertainment Conference (AIIDE)*, G. Sukthankar and I. Horswill, Eds. AAAI Press, 2013, pp. 100–106.

[54] H. Baier, A. Sattaur, E. J. Powley, S. Devlin, J. Rollason, and P. I. Cowling, "Emulating human play in a leading mobile card game," *IEEE Transactions on Games*, vol. 11, no. 4, pp. 386–395, December 2019.

[55] D. Churchill and M. Buro, "Hierarchical portfolio search: Prismata's robust AI architecture for games with large search spaces," in *Eleventh Artificial Intelligence and Interactive Digital Entertainment Conference (AIIDE)*, A. Jhala and N. Sturtevant, Eds. AAAI Press, 2015, pp. 16–22.

[56] T. Thompson, "Revolutionary Warfare — the AI of Total War (part 3)," https://www.gamasutra.com/blogs/TommyThompson/20180212/314399/Revolutionary_Warfare__The_AI_of_Total_War_Part_3.php, 2018.

[57] C. Browne, "Elegance in game design," *IEEE Transactions on Computational Intelligence and AI in Games*, vol. 4, no. 3, pp. 229–240, 2012.

[58] ——, "UCT for PCG," in *2013 IEEE Conference on Computational Inteligence in Games (CIG)*. IEEE, 2013, pp. 137–144.

[59] B. Kartal, N. Sohre, and S. J. Guy, "Data driven Sokoban puzzle generation with Monte Carlo tree search," in *Twelfth Artificial Intelligence and Interactive Digital Entertainment Conference (AIIDE)*, N. Sturtevant and B. Magerko, Eds. AAAI Press, 2016, pp. 58–64.

[60] B. Kartal, J. Koenig, and S. J. Guy, "User-driven narrative variation in large story domains using Monte Carlo tree search," in *2014 International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS)*, 2014, pp. 69–76.

[61] M. A. Pinheiro, J. Kybic, and P. Fua, "Geometric graph matching using Monte Carlo tree search," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 11, pp. 2171–2185, 2017.

[62] D. Hennes and D. Izzo, "Interplanetary trajectory planning with Monte Carlo tree search." in *Twenty-Fourth International Joint Conference on Artificial Intelligence (IJCAI)*, Q. Yang and M. Wooldridge, Eds. AAAI Press, 2015, pp. 769–775.

[63] A. Arora, R. Fitch, and S. Sukkarieh, "An approach to autonomous science by modeling geological knowledge in a bayesian framework," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2017, pp. 3803–3810.

[64] F. Golpayegani, I. Dusparic, and S. Clarke, "Collaborative, parallel Monte Carlo tree search for autonomous electricity demand management," in *2015 Sustainable Internet and ICT for Sustainability (SustainIT)*, 2015, pp. 1–8.

[65] A. Goldhoorn, A. Garrell, R. Alquézar, and A. Sanfeliu, "Continuous real time POMCP to find-and-follow people by a humanoid service robot," in *14th IEEE-RAS International Conference on Humanoid Robots (Humanoids)*. IEEE, 2014, pp. 741–747.

[66] G. Best, O. M. Cliff, T. Patten, R. R. Mettu, and R. Fitch, "Dec-MCTS: Decentralized planning for multi-robot active perception," *The International Journal of Robotics Research*, vol. 38, no. 2-3, pp. 316–337, 2019.

[67] Y. Qian, W. B. Haskell, A. X. Jiang, and M. Tambe, "Online planning for optimal protector strategies in resource conservation games," in *2014 International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS)*, 2014, pp. 733–740.

[68] T. Cazenave and S. B. Hamida, "Forecasting financial volatility using nested Monte Carlo expression discovery," in *2015 IEEE Symposium Series on Computational Intelligence (SSCI)*. IEEE, 2015, pp. 726–733.

[69] D. Bertsimas, J. D. Griffith, V. Gupta, M. J. Kochenderfer, and V. V. Mišić, "A comparison of Monte Carlo tree search and rolling horizon optimization for large-scale dynamic resource allocation problems," *European Journal of Operational Research*, vol. 263, no. 2, pp. 664–678, 2017.

[70] M. H. S. Segler, M. Preuss, and M. P. Waller, "Learning to plan chemical syntheses," *Nature*, vol. 555, no. 7698, pp. 604–610, 2018.

[71] F. Wimmenauer, "Monte-Carlo search for leveraging performance of unknown job shop scheduling heuristics," Master's thesis, Department of Data Science and Knowledge Engineering, Maastricht University, Maastricht, The Netherlands, 2019.