Autonomous Goal Selection Operation for Agent Based Architectures

Sravya Kondrakunta and Michael T. Cox

Wright State University, Dayton OH 45435, USA,
kondrakunta.2@wright.edu, michael.cox@wright.edu

Abstract. An intelligent agent has many tasks and goals to achieve over specific time intervals. The goals may be assigned to it or the agent may generate its own goals. In either case, the number of goals at any given time may exceed its capacity to act upon concurrently. Therefore, an agent must prioritize the goals in chronological order as per their relative importance or significance. We show how an intelligent agent can estimate the trade-off between performance gains and resource costs to make smart choices concerning the goals it intends to achieve as opposed to selecting them in an arbitrary basis. We illustrate this method within the context of an intelligent cognitive architecture that supports various agent models.

Keywords: Goal Operations, Goal Driven Autonomy, Cognitive Architectures

1 Introduction

It had long been recognized that intelligent systems need to do more than simply make plans to achieve their goals and then carry them out. Pollack and Horty [12] claim that much of what intelligent agents require in complex domains is not just planning for behavior but also that agents should manage their activities, resources, and their goals. In this spirit, a number of researchers have begun to examine intelligent and robust behavior from the perspective called goal reasoning [1, 8, 11]. In the most general sense, goal reasoning involves complex agents that can self-manage their desired goal states [15].

To perform such high-level goal management requires many cognitive processes and heterogeneous types of explicit knowledge. The kinds of cognitive and metacognitive processes that relate to this wide scope can be classified into a number of important categories. This paper examines a taxonomy of goal operations and looks closer at a particular example. Central to any agent trying to achieve multiple goals concurrently and/or sequentially is the process of choosing which goal or subset of goals to concentrate upon at any given time. Given that some goals are more important than others and that these priorities may change coupled with the fact that in high workload situations, an agent may not be able to plan or achieve all goals it currently has or may expect to get, hence goal selection is a crucial goal operation.

In the next section, we briefly describe a small set of important goal operations, and in the subsequent section, we examine a cognitive agent-based architecture within which most of these operations function. The next section then examines alternative ways to perform the goal selection operation including one based on expected performance over time. Experimentally we then evaluate and report results of our approach as compared to an uninformed approach that simply selects the oldest goal first. The next section, looks at related research, and we conclude the paper with a summary and future research directions.

2 Goal Operations

Goal reasoning has many characteristics, capabilities, and implications for advanced cognitive agents. Goals provide focus for inference and interpretation, and they provide a language for communicating intent to and between humans. To organize these various functions, we identify a set of principle goal operations that underlay much of the high-level cognition represented in many cognitive architectures. A minimal enumeration of operations is as follows.

- **Goal Formulation:** Generating independent, top level goals rather than waiting for external direction.
- **Goal Selection:** Committing to one or a number of goals from the complete set of pending goals by the agent based on some criteria (e.g., method reported below).
- **Goal Change:** Transforming the currently active goal or set of goals from one representation to another given a change of situation.
- **Goal Monitoring:** Paying attention to developing circumstances to notice when the reasons and priorities for goal achievement change.
- **Goal Delegation:** Giving a goal or a set of goals to another agent as circumstances dictate.
- **Goal Achievement:** Verifying that the goal state does indeed hold in the environment.

3 The Metacognitive Integrated Dual-Cycle Architecture

The *Metacognitive Integrated Dual Cycle Architecture (MIDCA)* is a high-level agent architecture with two cycles of processing. One is a cognitive cycle, and the other is metacognitive. Figure 1 shows details in the cognitive layer of MIDCA as an iterative repetition of processes together with an abstract representation for the metacognitive layer. The cognitive cycle perceives and interprets the environment then selects and executes object level actions (i.e., it would directly impact the physical environment). The metacognitive cycle performs introspective monitoring of the cognitive layer and performs control actions on the cognitive cycle.

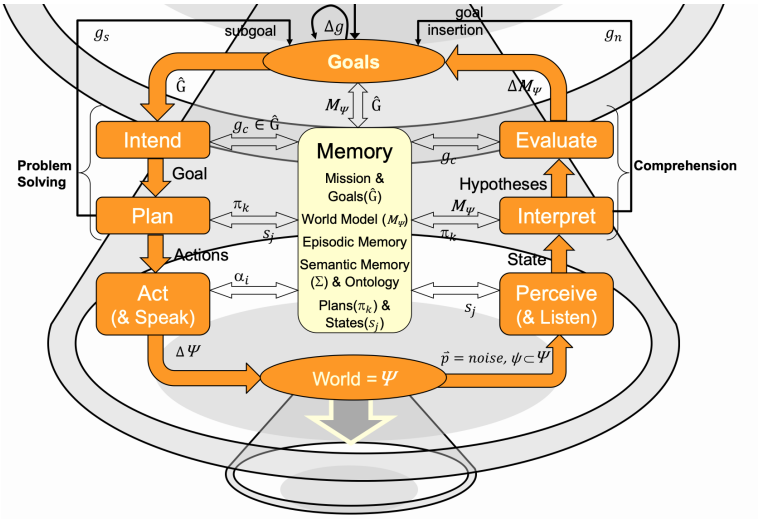

Fig. 1. Metacognitive integrated dual-cycle architecture. The cogni-tive cycle has six major phases with goal selection in Intend.

3.1 MIDCA Phase Transitions

Each layer in MIDCA has six phases and each phase performs its own unique functionality. The phases of the cognitive layer are as follows.

- **Perceive:** The perceive phase processes input from a subset of the environment $\psi \subset \Psi$ and updates changes to memory as a result. Percepts $\overrightarrow{p_j}$ can trigger existing perceptual monitor functions that match activation conditions (see below).
- **Interpret:** The interpret phase integrates percepts into a conceptual model M_Ψ of the environment Ψ that constitutes a hypothesis to explain the input, and it checks the interpretation for discrepancies that do not match expectations in the model. If discrepancies occur, a new goal g_n may be formulated to remove the discrepancy. Goal formulation adds the goal to a list of pending goals $\hat{G}$. A perceptual goal monitor is then created that will activate if the reason for g_n no longer holds.
- **Evaluate:** The evaluate phase checks for the satisfaction of current goal conditions g_c. Whenever the goal state holds in the interpretation of the environment M_Ψ, a goal achievement operation removes g_c from a MIDCA knowledge structure called the goal graph.
- **Intend:** The intend phase chooses a subset of goals from the pending goals $\hat{G}$ to become the new current goal condition g_c and inserts it into the goal graph. This goal selection operation is the main topic of this paper.
- **Plan:** The plan phase creates a new plan π_k for the selected goal g_c, unless a plan for it is already being executed. Although planning is not a goal operation, the planner generates perceptual plan monitors (similar to goal

monitors above) to detect future environmental conditions in ψ that would require plan adaptation (e.g., exogenous changes that correspond to operator preconditions).
– **Act:** The act phase executes individual steps α_i of the plan π_k. These executed actions will change the environment Ψ, and the cycle continues with Perceive.

The phases of the metacognitive cycle also work similarly to the cognitive phases, but the difference is that the meta-level phases interpret and act upon cognitive state rather than on environmental state. As each phase of cognition executes, MIDCA records an abstract representation of their operations in a declarative knowledge structure called a cognitive trace. The meta level "Perceive" phase then inputs the trace, passing on portions to interpret and look for discrepancies that do not meet cognitive expectations. Discrepancies at this level can result in meta-level goals to control the operations at the layer below. Execution of such meta-control actions can include the goal change operation. This directly changes goals in the cognitive cycle and indirectly changes action in the world.

3.2 The MIDCA Goal Graph

In MIDCA, goals can be classified as current or pending. The *current goals expression* (g_c) refers to one or more goals which the agent is committed to achieving. The *pending goals list* ($\hat{G}$) enumerates all goals including the current goal. These goals may be states that are externally given to it (e.g., from a user) or generated during a goal formulation operation. MIDCA represents the goal condition as a literal or conjunction of literals in first order predicate form such as the expression *on(A,B)*.

Here the symbol *on* is a logical predicate, and A and B are instantiated objects in the world. The literal can be either true or false. Actions change the value of literals when they execute. In MIDCA, goal formulation is done whenever an anomaly is detected, for example when a block catches fire, MIDCA generates a goal to put out the fire in the Interpret phase and immediately suspends the goal to have A on B. Once the goal to extinguish fire is achieved, the cognitive system resumes the stacking activity.

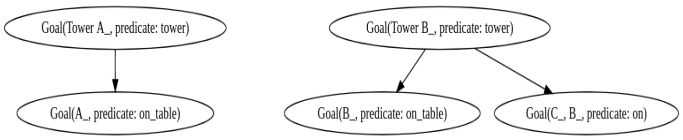

Fig. 2. Example goal graph structure

The *goal graph* is a representational structure for $\hat{G}$. The nodes in the graph form tree structures with the root node having the highest preference and its

child nodes having less. A goal graph might not always contain a single root; it might have multiple trees. The current goal expression g_c is stored in a separate list representing a subset of the goal graph

Figure 2 depicts a sample goal graph for the problem set of achieving two towers A and B of height one and two respectively. Here building a tower is a top-level goal and thus the root node for both goals. A root-level goal takes precedence over the sub goals on-table and on. The achievement of all the sub goals listed under the higher level goal indicates the achievement of the higher level goal.

The MIDCA phases interact with the goal graph in memory. Table 1 enumerates the kinds of ways each phase in the cognitive layer uses or manipulates the goal graph.

Table 1. Interactions between cognitive phases and the goal graph

Module	**Interaction with Goal Graph**
Perceive	No interaction.
Interpret	Gets the goals from the user and inserts them into goal graph.
Evaluate	Checks to see if the current goal or goals are achieved and if so removes the goal or goals and its corresponding plan from the goal graph.
Intend	Checks to see if the goal graph is empty, if yes skips else checks if current goal is empty, if yes then selects the goal based on strategies like FIFO or information measures and inserts into current goal. If no, then skips. Intend also inserts the formulated goal when an anomaly is detected and places it above the root node to give it highest priority.
Plan	Checks the goal graph for a matching plan, if exists, it checks validity. If no matching plans or plans are not valid, generates a new plan and inserts into goal graph.
Act	Iterates over the plan for current goals in order to achieve it.

4 Goal Selection with a Selection Criterion

Goal selection with a selection criterion has been motivated by goal reasoning with Information measures [9]. The search goal is implemented in surveillance of three different regions, airport and two office buildings to locate an official in which the author uses distance traversed and time taken to calculate the information gain. To implement the goal selection in MIDCA we have considered two factors, score and limiting factor which are domain specific. The score is used as an estimate of performance or benefit and the limiting factor is used as an estimate of cost. The goal selection operation is achieved through processing information obtained from the factors considered. The information might be something which the user thinks to be appropriate for the selection procedure, in the described domain a scoring or performance function and time has been used for performing the selection operation. Rather than selecting the goals randomly or by a First-In-First-Out (FIFO) basis, the selection of goals by using some factors from the domain appears more sensible, intelligent and shows improvement in performance. If we have all the amount of time or no limitation on time then the goal selection would not have been an operation to consider and implement,

as all the goals would have been achieved by any of the methods, but in the real world that surely is not the case as we do not have all the time. To understand the functioning of the algorithm the domain description is presented in the next section and then algorithm is explained with an example in experimental method and finally the results are evaluated.

5 Domain Description

The goal selection operation has been implemented in a construction domain. The domain is named so because the goals generated are to build towers. This domain is an extension to the simple blocks world domain. The goals to construct the towers are generated randomly, number of towers in each problem set will be a random number between one to seven and the height of the towers vary from one block to seven blocks as well. All the goals in a single random set might or might not be distinct in height but would be distinct in objects i.e., towers of same height will can be generated in a single random set but say if a block named 'A' is used in one tower then it would not be used in a different tower of the same problem set. The above is implemented in order avoid the process of demolition to construct a new tower within the same set of goals.

Initially all the blocks are kept at the warehouse and the construction site would be empty, the user can see the construction site but the warehouse is invisible. So whenever a problem set is generated, the agent selects all goals to be achieved by calculating the ratio of the estimates of factors considered for the domain. The objects related to the relevant goals will be fetched one at a time from the warehouse and a corresponding operation will be performed on each object. The problem set is generated every time when MIDCA is initialized and the selection process is performed on every problem set, the towers constructed previously would be erased each time when a new problem set arrives.

The operators described in this domain include *stack, stack_mortared, unstack, unstack_mortared, pickup, put_down, get from_warehouse, put_out_fire.* Each operator is unique and performs different actions. The *stack* operator places one block over the other block. The *unstack* operator removes one block from the other block. The operator *stack_mortared* does the same action as *stack* but with mortar, the operator *pickup* block is functional only when the block is on ground/table. The *putdown* operator is executed when the block selected should be placed on ground/table and finally the *get_from_warehouse* operator is used to get the objects from warehouse to site.

Some of the predicates include *clear, on, on-table, in-warehouse, stable-on.*

- *clear(X)*: nothing above object X
- *on(X, Y)*: object X is on Y
- *stable_on(X, Y)*: object X is on Y with mortar
- *in_warehouse(X)*: X is in warehouse

There are a total of twenty eight objects named A-Z, Z1, and Z2. All the objects are clear and in warehouse at the beginning, based on the goal set

generated the blocks are transported from warehouse to the location using the *get_from_warehouse* operator.

5.1 Experimental Method

Each time MIDCA is initialized, a random goal set to construct some n number of towers is given, and MIDCA performs goal selection from the input goals using a FIFO or the selection criterion. Under FIFO for a randomly generated list of towers, the first goal is selected and achieved and then the next and so on. For the method using factors from the domain, MIDCA chooses based on a decision ratio between estimated score and time.

Performance Function A scoring function assigns each tower a performance number when the construction of a tower is completed within the time constraints. Each tower would be constructed by stacking n number of blocks, for a tower of height h constructed successfully within the deadline the score achieved would be h. And the score for constructing m number of towers whose height ranging from $h_1, h_2, \ldots h_m$, the score would be $P = \Sigma_{n=1}^{m} h_n$. The towers which exceed the deadline will receive a score of zero. No partial scores are assigned for the towers which are constructed partially within the deadline.

Cost Function The construction of any tower takes time, and the time taken would increase as the height of the tower increases. There will not just be an increase in overall time, but there will also be an increase in the time for each stack as the height of the tower is increased. As such, the increase would be a nonlinear function. In the implementation, the time values are provided as estimates for the agent. As the height of the towers vary from one to seven, for each tower an estimate of time is provided as shown in Table 2.

Table 2. Estimated times $\hat{t}$ to construct each tower and estimated time to place upper block in each tower

Tower height	Overall Time	Time to place upper block
1	1	1
2	2.2	1.2
3	3.4	1.2
4	5.4	2
5	8.4	3
6	13.4	5
7	22.4	9

Now the ratio C of the performance function $\hat{P}$ over the estimated time $\hat{t}$ is calculated for all the goals in the random problems, and it acts as a selection criterion for goals. In this manner, the tower with the maximum C criterion is chosen first.

$$C = \hat{P}/\hat{t} \tag{1}$$

This goal is achieved, then the goal with the second highest ratio is selected and achieved and so on. In this particular domain, it does not matter if we start the construction with either a maximum or a minimum ratio of tower. The maximum ratio indicates a tower with the minimum height in the problem set, and the maximum ratio indicates the tower of lowest height. we choose to start with maximum ratio because, it yields best output per unit time. To check the performance of the above function for various scenarios the deadline can be varied dynamically. The variation can be in either smaller or large amounts, if the variation is least then choosing minimum ratio would be good as towers with smaller height can be dropped but if the deadline is varied by large amounts, then the agent can drop the goal it is currently working on and go with the small towers when the time is sufficient.

The evaluation method functions by taking into consideration four cases. One is with no deadline, and the others are with a particular deadline. Let D indicate the overall deadline or deadline for a particular problem.

Case 1: No Deadline In this case, as there is no deadline (i.e., $\boldsymbol{D} = \infty$), all the towers in each random problem will be constructed one after the other. The agent would achieve all of its goals, and therefore the score achieved would be 100%.

Case 2: Deadline of D=X In this case, as there exists a deadline of X, all the towers within the problem set may or may not be constructed before the deadline. The algorithm must be able to choose the best possible subset of goals which can be achieved from the problem set within the deadline of X. The combination of all the goals within X is listed, the summation of scores within each combination is also listed and the one with maximum score is selected.

Consider a simple problem set of three towers B_1, B_2, B_3 with the estimated scores $\hat{P_1}, \hat{P_2}, \hat{P_3}$ and estimated times being $\hat{t_1}, \hat{t_2}, \hat{t_3}$ respectively, the deadline being the same X. Assume that $\hat{t_1} \leq X, \hat{t_2} \leq X, \text{and} \hat{t_3} \leq X$. The tower B_3 is eliminated in the first place itself as the time taken to construct the tower exceeds the limit. Now consider the other two towers and check if $\hat{t_1} + \hat{t_2} \leq X$if yes then the possible combinations of towers would be $B_1, B_2, B_1 + B_2$ and their respective scores are either $\hat{P_1}, \hat{P_2}, \hat{P_1} + \hat{P_2}$ the greatest among the three is surely $\hat{P_1} + \hat{P_2}$. So the two towers B_1 and B_2 are constructed. Among those, as only one tower can be constructed at a time the one with highest $\hat{P}/\hat{t}$ is selected. Else if $\hat{t_1} + \hat{t_2} > X$ then the possible combinations are B_1, B_2 and the tower with maximum score among the two is selected and constructed.

Case 3: Deadline of D=X and varying t and P In the previous case, construction strictly follows the expected score and estimated time but in reality the situation is never the same, it varies, so a random function is introduced

into the scenario, a seed is programmed to the random function to make the randomness of FIFO and selection method uniform. This random function would vary the score, time obtained at each action within ±20%, ±50% range of the defined score and time values or expected values. For example, assume that the score assigned for a successful operation is one then ±20% variation in score refers that the score might be any random value between ±20% of one, or it can be any random value between 0.8 to 1.2.

Case 4: Varying the Deadline, t and P In this case, the deadline varies within the range of ±50% of D i.e., the deadline can be further increased or decreased dynamically as the construction is in progress or while the goal is being achieved. Along with the deadline, the score and the time are also varied in the range ± 20%, ±50%.

6 Evaluation Results

The evaluation of the goal selection operation has been done by comparing the percentage of the scores achieved by FIFO and the goal selection based on the ratio of factors considered. Even this evaluation has been split into the same two cases. For the first case i.e., the one with no deadline, even if both methods choose the goals in different orders, the graphs coincide as 100% is achieved by both methods. For the second case which is when a deadline is introduced, the evaluation is performed for two different deadlines 5 and 10. The normalized cumulative score is calculated for both the scenarios and is plotted against each problem set. On the X-axis each problem set is the average of three different problem sets and the Y-axis presents theirs normalized cumulative score. So as a whole 30 problem sets are considered and represented in the graphical format. The normalized cumulative score is calculated by comparing the ratio of score achieved against the overall score i.e., consider the case with $D = 10$ and the problem set is to construct the towers of height 4, 2, 5, 3. In the mentioned problem set the score obtained by FIFO is 6 (as FIFO constructs the towers of height 4 and 2)and the score achieved using the selection method is 7 (as the Selection Method constructs the towers of height 4 and 3). Now their respective score percentages would be 6/14=0.428 and 7/14=0.5. The percentages obtained for 3 problem sets are taken and averaged to represent as a single problem set.

Figures 3 and 4 depict the difference in the performances of FIFO and the goal selection with the selection method, in both the cases it is very clear that the method using factors from domain outperforms FIFO, even with a larger deadline the two methods might obtain same percentage in some cases but the FIFO never yielded a better result than the selection method when every problem set follows the expected values.

There exist some problems like problem set 3 in Figure 3 and problem set 8 in Figure 4 where the performance of both the selection method and the FIFO coincide i.e., a parallel line indicates the value to be same but this coincidence does not necessarily mean that all the result of all the three problem sets coincide.

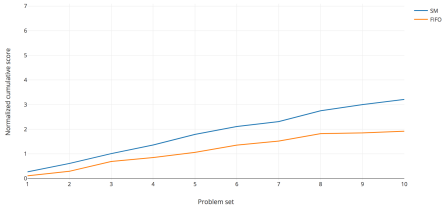

Fig. 3. *Cumulative SM and FIFO scores with* $D = 5$

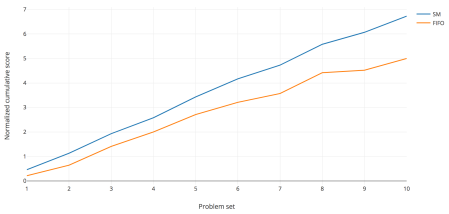

Fig. 4. *Cumulative SM and FIFO scores with* $D = 10$

There are also cases like a problem set has two towers to construct then both the goals exceed the deadline and nothing would be selected by intend. In that case the score achieved would be zero. When the graphs in Figure 3 and Figure 4 are compared it can be easily observed that the percentage of goals achieved when the deadline is increased is higher, this is because of the fact that there is sufficient amount of time present to achieve more goals.

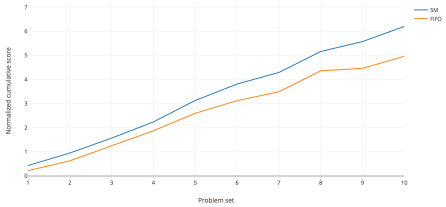

Fig. 5. *Cumulative SM and FIFO scores with* $D = 10$ *and both* P *and* t *varies by* $\pm 20\%$

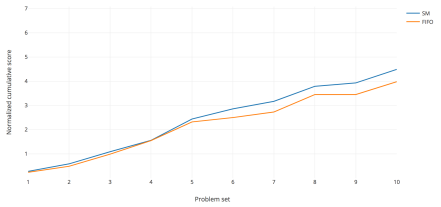

Fig. 6. *Cumulative SM and FIFO scores with* $D = 10$ *and both* P *and* t *varies by* $\pm 50\%$

Figure 5 describes case 3 where the score and time varies by either $\pm 20\%$ or $\pm 50\%$ and the deadline remains constant and the actual scores with the selection method and FIFO are plotted. In this case the time and score are varied within $\pm 20\%$ range of the expected values and the deadline is 10 and it remains 10 throughout, if for an action say the time taken would be 1, then the range of the actual time would be within 0.8 to 1.2 and any number between the range could be the time taken/actual time to complete the action. The variation is similar with the scoring function as well. The problem sets 3 and 8 have the same actual scores for the selection method and FIFO. The actual efficiency obtained in this case using the selection method is around 60% whereas for FIFO it is around 50%. Even in this scenario FIFO is either lesser than or equal to the selection method because the variation in the actual values from the actual values is around 20%.

Figure 6 describes the case where the deadline is constant value 10 but there is a variation in score, time values by ±50%. The actual scores for both the methods are plotted in Figure 6. As we can notice the method using the selection criterion yielded better results than the FIFO but if we take a closer look there are also some cases where the FIFO performed better than the selection method. Examples include problem sets 4 and 8.

Consider problem set 8. It is the set of three different problems generated randomly and they are problem set 1: Towers of height 3, 2; problem set 2: Towers of height 4, 1, 2; and problem set 3: Towers of height 2, 2, 3, 6. For the above problem sets the selection method constructed the following towers: problem set 1: 2, 3; problem set 2: 1, 2; and problem set 3: 2, 2; whereas the FIFO constructed: problem set 1: 3, 2; problem set 2: 4, 1; and problem set 3: 2, 2. The time and performance in this case is varied by ±50% of the expected time, the values for of actual time and scores for the above problem sets for FIFO and selection method are as following: For FIFO: t, P for problem set 1: (7.48, 5.412); problem set 2: (7.95, 5.516); and problem set 3: (5.75, 3.956); For selection method: t, P for problem set 1: (7.48, 5.412); problem set 2: (4.133, 3.322); and problem set 3: (5.75, 3.956). There is a lot of variation in problem set 2 than the other cases as the selection method started off by constructing the tower with maximum ratio but the actual time and score varied drastically than expected and it failed, whereas the FIFO started with the tallest tower as it is generated first in the problem set and got a highest score irrespective of the variation. So, when the variation in the performance measures is very high than the expected the selection method might not work as best as expected. If we take a look at the overall efficiency, then selection method is a little greater than 45% and the FIFO is around 40%.

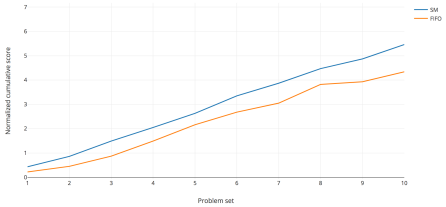

Fig. 7. *Cumulative SM and FIFO scores with $D = 10$(but varies by ±50%) and both P and t varies by ±20%*

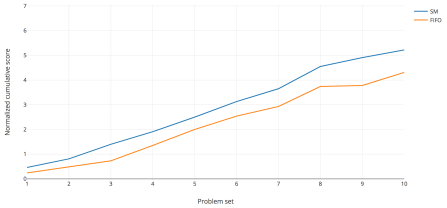

Fig. 8. *Cumulative SM and FIFO scores with $D = 10$(but varies by ±50%) and both P and t varies by ±50%*

Figure 7 describes the case where the score and time vary with a varying deadline. The actual scores with selection method and FIFO are plotted. In this case the time and score are varied within ±20% range of the expected values. As we can observe the variation of the deadline is also introduced in this case and it is with ±50% range of the expected values. The efficiency using the selection

method is around 55% and for FIFO it is a little less than 45%. Even in this case FIFO beats the selection method at problem sets 3, 4 and 8, but the amount is very small. Even in this case the reason for the selection method not performing as good as expected is similar to the previous, i.e., if the variation in actual values is very high than expected then the selection method might not work good for some cases.

Figure 8 depicts the case for varying the deadline, score and time by ±50%. Even in this case selection method yielded better results than the FIFO but in some cases like problem set 4 and problem set 10 the FIFO yielded better results than selection method. The efficiency of selection method is greater than 50% and for the FIFO it is a little greater than 40%.

7 Related Research

Goal selection from a given set of goals by assigning priority values to each goal has been presented in the ICARUS architecture [3]. The range of the priority values vary from zero to ten with zero being the minimum and ten a maximum value. Goal Reasoning with Informative Measures (GRIM) [9] is the system in which some selection metrics like distance traversed and time are used to solve the goal selection problem in a specific domain. This work is taken and has been modified to fit in the blocks world domain in MIDCA architecture. In [16], Goal Driven Autonomy is applied to underwater unmanned vehicles where the vehicle is left to explore places that are undesirable for humans, in such places it is very important for the vehicle to formulate prioritize and assign the goals in a dynamic environment. All the works states above perform the goal selection using various different methods.

The goal operations work is not only limited to the cognitive systems but it is also useful in other applications like space research [2], here the new goals are triggered based on the outcomes of the previous goals. This work has motivated Rabideau et al. [14] in which they develop a goal selection algorithm with oversubscribed resources by considering the constraints and priorities to choose a goal.

The works of Klenk et al. [10], Cox [4] and Dannenhauer and Munoz-Avila [6] on goal driven agents generation of new goals when a discrepancy is detected has been a good motivation for the goal formulation described in the paper. Work on T-ARTUE [13] learns the knowledge of goal selection from a user and the criticism provided by the user for wrong selection, through interactive learning. Harland et al. [7] provides various semantics for the goal life cycle in BDI agents, and different types of goal states are described. It specifies the goal management in an agent and also verifies for its correctness, this paper does not focus on multi agents. Cox et al. [5] discusses the idea of goal transformations and provides a formalism for several goal operations including goal selection.

8 Future Work and Conclusion

Goal selection is an action which humans perform on a regular basis. For example, selecting the type of food, clothes and so on. Everyone performs those operations based on their own preferences, resources available and expectations. However, in this paper the goal selection is implemented using an intelligent selection criterion on a simple scenario. This work can be further extended when the construction is implemented with an extra resource called mortar. The purpose of the mortar is to make the tower sturdy, i.e., if a tower is constructed with mortared blocks then the stability of the tower would increase and the score increases for each block while stacking as opposed to the regular stacking (presented in the paper) which would be useful when there are not enough mortar, and the constructed tower will not be as sturdy as mortar tower hence the score would remain the same. As mentioned in the paper, anomalies like fire can occur while construction, in such a scenario when an arsonist sets up a block on fire, the priority should be given to the goal that puts the fire out rather than continuing with the construction. The selection strategy should be extended to include anomalous scenarios.

The work can be further generalized to other domains. For example, the restaurant domain where the score can be an estimate of satisfaction of the customer and the limiting factor can be money. The actual vs expected scores can be compared by introducing a variation similar to the construction domain. The goals presented in the paper are currently static and do not change, there is also scope for the work to improve when the agent is encountered with scenarios where the goals change dynamically. Complexity analysis on the work is not done yet but can be worked on in future. The goal selection and goal transformation can be applied together when the agent is out of mortar, then it can either quit or the goal of "stable-on" can be transformed to "on" and then continue with the construction. The work shows a clear improvement than the previous FIFO method.

Acknowledgements

This research was supported by NSF under grant 1849131 and by ONR under grant number N00014-18-1-2009. We thank the reviewers for the comments and suggestions.

References

1. Aha, D. W., Cox, M. T., Muñoz-Avila, H.: Goal Reasoning: Papers from the ACS Workshop (Technical Report CS-TR-5029).University of Maryland, Department of Computer Science (2013)
2. Chien, S., Cichy, B., Davies, A.: An autonomous earth-observing sensorweb. IEEE Intelligent Systems 20(3)16 – 24 (2005)
3. Choi, D.: Reactive goal management in a cognitive archi-tecture. Cognitive Systems Research 12(3): 293-308 (2011)

4. Cox, M. T.: Goal-Driven Autonomy and Question-Based Problem Recognition. In Proceedings of the second Annual Conference on Advances in Cognitive Systems, 29-45. Palo Alto, CA: Cognitive Systems Foundation (2013)
5. Cox, M. T., Dannenhauer, D., Kondrakunta, S.: Goal operations for cognitive systems. In Proceedings of the Thir-ty-first AAAI Conference on Artificial Intelligence, 2501–2507. Palo Alto, California: Association for the Advancement of Artificial Intelligence (2017)
6. Dannenhauer, D., Munoz-Avila, H.: Raising Expec-tations in GDA Agents Acting in Dynamic Environments. In Proceedings of the International Joint Conference on Artificial Intelligence. Palo Alto, CA: AAAI Press (2015)
7. Harland, J., Morley, D. N., Thangarajah, J., Smith, N.Y.: An operational semantics for the goal life-cycle in BDI agents. Autonomous Agents and Multi-Agent Systems 28(4):682-719 (2014)
8. Hawes, N.: A Survey of Motivation Frameworks for In-telligent Systems. Artificial Intelligence 175(5): 1020-1036 (2011)
9. Johnson, B., Roberts, M., Apker, T., Aha, D.W.: Goal reasoning with information measures. In Proceedings of the Fourth Conference on Advances in Cognitive Systems. Evans-ton, IL: Cognitive Systems Foundation (2016)
10. Klenk, M., Molineaux, M., Aha, D.W.: Goal-Driven Autonomy for Responding to Unexpected Events in Strategy Simulations. Computational Intelligence 29(2): 187-206 (2013)
11. Paisner, M., Maynord, M., Cox, M. T., Perlis, D.: Goal-Driven Autonomy in Dynamic Environments. In D. W. Aha, M. T. Cox, H. Munoz-Avila (Eds.), Goal Reasoning: Papers from the ACS Workshop, 79-94. Tech. Rep. No. CS-TR-5029, Department of Computer Science, University of Mary-land, College Park, MD (2013)
12. Pollack, M., Horty, J.F.: There's More to Life Than Making Plans: Plan Management in Dynamic, Multiagent En-vironments. AI Magazine 20(4): 71-83 (1999)
13. Powell, J., Molineaux, M., Aha, D.W.: Active and Interactive Discovery of Goal Selection Knowledge. In Pro-ceedings of Twenty-Fourth International Florida Artificial Intelligence Research Society Conference, 413–418. Palm Beach, Florida: Association for the Advancement of Artificial Intelligence (2011)
14. Rabideau, G., Chien. S.: Runtime Goal Selection with Oversubscribed Resources. In Proceedings of Interna-tional Joint Conference Artificial Intelligence Workshop on Oversubscribed Planning. Chicago, Illinois: Association for the Advancement of Artificial Intelligence (2008)
15. Vattam, S., Klenk, M., Molineaux, M., Aha, D.W.: Breadth of Approaches to Goal Reasoning: A Research Survey. In D. W. Aha, M. T. Cox, H. Munoz-Avila (Eds.), Goal Reasoning: Papers from the ACS Workshop, 111-126. Tech. Rep. No. CS-TR-5029, Department of Computer Science, Uni-versity of Maryland, College Park, MD (2013)
16. Wilson, M., Auslander, B., Johnson, B., Apker, T., McMahon, J., Aha, D.W.: Towards Applying Goal Autonomy for Vehicle Control. Goal Reasoning: Papers from the ACS Workshop, 127–142. Department of Computer Science, Uni-versity of Maryland, College Park, MD (2013)