The CSI Journal on
Computer Science and Engineering
Vol. 2, No. 2&4 (b), Summer & Winter 2004
Pages 41-51
Regular Paper

Going Meta: Back to the Expectations

Shahriar Pourazin Ahmad Abdollahzadeh Barforoush

Computer Engineering Department, Amirkabir University of Technology
Tehran, Iran

Abstract

It is time to use the great implementation achievements and have a look back at the ideas which were treated as pure theory. One of these ideas concerns with metareasoning and it has been under focus in the rest of paper. A classification for types of metareasoning has been proposed. In recent years, only (the ones that here named as) pre-metareasoning and para-metareasoning have been studied. The first one is for predicting the best computation path for having better performance programs. The second, mostly known as interruptible anytime algorithm, is to limit the computation time externally when the approximate answer is better than nothing. One other type of metareasoning (called here as post-meta reasoning) is discussed in a case study. It has been shown as an effective method for reducing error in self-localization. Post-metareasoning argued as useful when the effectiveness of reasoning methods are not known.

Keywords: metareasoning, para-metareasoning, pre-metareasoning, post-metareasoning, self-localization

1. Introduction

AI researchers worry about the future of AI. Among them there is an idea that current AI is not that great to be memorable in, say, 200 years time [31]. Current advances in AI are not as expected. The recession in AI (if not caused by) is in parallel with conservative approaches to enhance the existing ideas. Saul Amarel (in [21]) comments that: "In several parts of the field, the emphasis is on the refinement rather than on development of new ideas or exploration of new territory." Here, in this paper, we try to define a taxonomy of types of metareasoning. The proposed classification opens a new research frontier in the scope of one possible class of metareasoning which (up to our knowledge) has not been noticed yet. To clarify the matter, a case study is presented. The case study is by no means, supposed to prove the ideas of paper empirically.

The history of AI shows that, early AI focused on extremely ambitious goals while more recent AI has focused on smaller, more attainable goals. Now, it is time to return to more ambitious goals, and that metareasoning is one such goal. It is declared that, the existing insights of reasoning are insufficient to model all aspects of intelligence [43]. The very complicated intelligent activities like creativity is claimed to be possible through metalevel reasoning [5].

Metareasoning is reasoning about reasoning and not about the external objects. The term "metareasoning" is defined as the meta-level deliberation about internal objects such as computations and beliefs and in contrast with object-level deliberation about external entities (such as another agent) [42]. In other words, it is planning about reasoning and not planning about external activities [45].

The proposed taxonomy of meta-reasoning has three elements: meta-reasoning before, during, and after domain-level reasoning. Maybe the most attractive type of metareasoning is the one that is done before the low level reasoning. It can save the resources of the intelligent system better than other types. There are sorts of meta-reasoning to be done in parallel with problem space reasoning. This kind of metareasoning is useful especially when the intelligent system is still searching for new reasoning methods and it is not wise to stop the existing domain-level reasoning. Here we see that in a case, it is still useful to continue

metareasoning even after completion of one, two or more reasoning procedures. The experiment involves combining results from different known techniques for (simulated) robot self-localization. Specifically, each technique is employed separately, and the one result with the smallest total distance to all of the other results is selected. Experiments show that this mechanism outperforms all of the competing mechanisms proposed for combining results of different techniques.

In section 2, first, we have a look at the trends in the modern life of AI. This is to support the idea of returning to early complicated AI goals. In next section, metareasoning will be described as an important topic deserving further research. Next two chapters will include the classes of metareasoning and a more in-depth look at one of these classes called here as: post-metareasoning. Conclusion is the last section and includes the results and some proposed topics for further research.

2. Trends in AI: from Past to the Future

It has been claimed that between 1950s and 1970s, there has been a trend in general problem-solving by theoretically complex and even imaginary systems. Then, from 1970s to 1995, there was a time that researchers have a major shift to performance programs "whose performance was superior to that of any human not having specialized training, experience, and tools. And after 1995, there should be another transition to reinvigorate e orts to build programs of general competence" [36]. Here we adopt the Nilsson's prediction. We just extend his ideas slightly to split the age of performance programs into two periods of experimental and critical applications.

2.1 Early days of pure ideas

The impact of newly invented computers, has been rejected in the first ideas which were too general and theoretical. The ideas were general, because of having no special field of application and covering all aspects of intelligent behavior. They also were theoretical, because there was no clear implementation plan to achieve the final specifications. Turing and his friends talked about their astonishing "child machine". This conceptual machine was supposed to be implemented by discovering the internal processes that brain acquires its intelligence. So, it was called teachable intelligent machine [32]. "Automatic computers", as they have been referred at that time, were evaluated as having only the problem of speed and memory to simulate many higher functions of human brain. And except this, it was our inability to write programs taking full advantage of what we have [29].

The terms like "general intelligent action" (in the definition of physical symbol system hypothesis [34]) and "all purely intellectual fields" (in the Turing's speculations [51]) are used as general specifications. To achieve the "general intelligent action" we only knew that we have all "necessary and sufficient means" [34] but we didn't know clearly from where to start. And also there was a claim that "machines will eventually compete with men" [51]. And again how to make such a machine, was not clear and there was almost nothing but theoretical claims.

It then became clear that general intelligence will not be achieved immediately. The limited specification intelligent systems appear. Most people forgot the excitement of heroic ideas. Among the reasons for this withdraw, Aaron Sloman mentions some which seem quite important [48]:

- *People did not understand the difficulty, complexity and variety of the (intelligent) tasks.*
- *They did not know enough philosophy, linguistics, psychology, biology.*
- *It was too easy to assume that we already know what humans can do, leaving only the task of modeling and explaining human capabilities.*
- *Ideas about software architectures were still too primitive. Computers were pathetically inadequate for many years (except for limited classes of tasks).*

2.2 Rush to make high performance programs

Making general intelligent systems has changed to making limited intelligence for special purposes. Like making an expert system to implement a limited subset of human expertise for a special problem area. In a major report [53], "two main areas of AI" were determined as expert systems and natural language processing in next 5-10 years. The impact of strategic computing program at ARPA in '80s was the development of down to the earth AI applications.

This can be treated as the start of recession age and disappointment in the way toward general intelligence. Sir Lighthill's report [26] in early 1970's evaluated AI as failed. His criticism on combinatorial explosion has been discussed in [30] as what AI should avoid by heuristics. Since then, the termination of inference in a limited time has been considered while talking about expert systems and other intelligent systems.

2.3 Critical applications age

This period is the start of an age in which the critical applications were designed to let people rely on them. The di erence of these ages with the previous one, is in the number of problem areas addressed by AI systems. The use of intelligent systems dominated. We still see that the implementation reports and experiment results are the hottest topics to be taken into consideration. And the main focus is still on performance programs. The start of this age is something around the time when the report about the ARPA's twenty-first century intelligent systems [17] published.

This was the relief age. People started to forget the severe criticisms about AI. Confidence seemed to come back to community, and empirical methods had a major role in that [52]. More powerful computers and success stories led the researchers to explore new directions, talking about the theory again. It was time to look back to meta level reasoning.

2.4 Back to future: ideas

We are now in the starting years of the new age. We again consider the old ideas of more general intelligent systems. Again, AI concerns the relationship between artificial intelligence and human intelligence. Exactly the same way that we compare flying machines with birds [21]. The robots are aimed to win humans in a real soccer game around 2050's (see www.robocup.org). "We are right at the start of adolescence" [1]. So, again we can think about programs of general humanlike competence [36]. "The long-term goal of emulating general human intelligence remains,

and that goal will bind future generations of AI researchers" [31].
Performance programming will make the necessary tools for the complicated systems. We should use AI techniques to design new systems for old and complicated problems. Some scientists believe that the grand challenges for the future (21st century) are more of scale of design than scale of computation [35]. To follow this idea, we should design the systems with different concepts.
We have to remember what has been forgotten for years. We should face the problems like metareasoning.

3. Why Metareasoning?

Until 1990's, there was a long time that metareasoning was forgotten: Early in the '80's *there was much hope that many hard problems could be solved by "going meta." So far, meta-reasoning has not turned out to be a panacea [4]*. After early efforts in development of process (reasoning and metareasoning), some researchers switched to content (knowledge) and this has been done due to weakness of general problem solvers and success of powerful expert systems(see comments by E. A. Feigenbaum in [21]).
Cyc [23] was an e ort to show that AI systems do not need the sophisticated reasoning mechanisms. Only modus ponens is enough. We only need more (common sense) knowledge. It is obvious that, having huge amounts of (domain) knowledge, will make the system more powerful. But there are cases that all aspects of intelligence can not be implemented just by more object level knowledge. When trying to simulate fast and correct but imprecise responses of human, we have to make use of other types of inference like abduction and approximate reasoning. Metareasoning is a tool to select the appropriate inference method, based on the constraints imposed by environment. Metareasoning is even necessary when we have more than one inference method to choose.
If an agent is unable to perform the rational action due to limited time, the agent is facing the problem of bounded rationality [16,8,47]. Metareasoning is really what the agent can do to select the best deliberation action to minimize the required time and have the rational response. This way, metareasoning tackles the combinatorial explosion and this approach has been more or less survived during recent years from early 1990's.
Oliver Selfridge, in his comments about the greatest trends and controversies in AI [21] says: "Find a bug in a program, and fix it, and the program will work today. Show the program how to find and fix a bug, and the program will work forever". He has mentioned that "AI software should be more concerned with being changeable - and all that that implies." In his opinion, machine learning, along with neural nets and genetic programming are tools for building such an AI software. Here we accept the importance of the topic and argue that, metareasoning could deal with the same important problem of changing the programs by themselves especially when we want to make the system evaluate the modified parts of code in itself. This approach to metareasoning (compared with bounded rationality case above) does not necessarily control the combinatorial explosion.

4. Proposed Classes of Metareasoning

Basic metareasoning problems have been classified as: (1) defining shares of time in which each of the anytime algorithms run, (2) dynamically allocating evaluation e ort across actions, and (3) dynamically choosing how to disambiguate state [7]. In the first type, the agent should do both the actions, say, A and B. It can assign different processing times to each of actions A or B. Means that the actions are handled by anytime algorithms [11] and could give better results if they have more time. Metareasoning in such a case, will help the agent to decide on how much time to allocate for each action.
In the second type, there are (say) three equivalent actions A and B and C to achieve a unique goal. Only one of them is sufficient to be selected by the agent. The agent needs to evaluate all three actions, and has time to evaluate only two of them. Otherwise there will be no time left for taking one of the actions.
In the third category of basic metareasoning problems, the agent should do all actions but just one of the possible orders of precedence among A, B and C will be acceptable. Consider that doing B before doing C may be fatal. In this situation, metareasoning is to define the order to do A, before or after B, before or after C.
In our approach we categorize all three above types of metareasoning in one class called here as pre-metareasoning. The reason is simple. In all cases above, we are doing metareasoning before starting the reasoning (action) itself. It is obvious that the metareasoning could be done in parallel with reasoning specially when we have interruptible anytime algorithms [44]. And also we may have post-metareasoning which is the major topic covered in the case study section of this paper (see section 5).
There are also other ideas that declare meta level reasoning as an instance of the metalevel programming. A meta program is a program that processes another program. A compiler is an example of a meta program. When the meta program runs in parallel with the processed program, the system should switch between meta level and object level computation repeatedly. When a system switches between meta level and object level processing, it exploits reflection [10]. There are implementations of reflection like FOL [54], Omega [2], 3-Lisp [49] and Re ective Prolog [9]. MetaProlog [3] was an e ort to amalgamate the meta-level and lower level languages. FreshML [46] and Elf [37] are e orts in meta-programming. GETFOL [15] is a mixture of 3-Lisp and FOL. GETFOL is intended to give the theorem-provers, the ability of implementing flexible control strategies to be adapted to the particular situation by reflection. ObjVProlog [27] is a language which lets its objects to be modified by the use of a "meta-object protocol" which affects meta-objects. Flexible Inference Engine [40] is designed to amalgamate the knowledge and meta-knowledge and to process both in the same manner. It uses Lisp as its knowledge representation language so it represents the programs and data structures beside the meta-programs in Lisp within its knowledgebase (as knowledge and meta-knowledge).
A more complete taxonomy of metareasoning covers more dimensions of metareasoning beside the time when the metareasoning occurs. One may classify metareasoning by

considering its role to increase the speed of reasoning [33], to improve the quality of results (see the case study in section 5), and to change the functionality of the reasoner [20, 50]. These aspects are related to and evaluated by the outcome of metareasoning. Although there could be lots of characteristics to be used to classify the types of metareasoning, here, the interaction and precedence of levels of reasoning are under focus. The impact of this approach can be directly translated to the design constraints in the architecture of the intelligent system: the intelligent system should (1) have reasoning and metareasoning as concurrent processes, (2) have metareasoning as the parent process of the reasoning process(es), (3) continue metareasoning for ever from the time before reasoning, during the reasoning and after reasoning.

The proposed classification in this paper is not in contrast with any existing classification of metareasoning systems. But this approach, could let us better locate the places where metareasoning can be settled in the design of intelligent systems. It also helps us accept new methods of existing processes and choose among them on the y. If the choosing procedure is time consuming, we can postpone the selection, to be in parallel and/or after reasoning. This way, we reduce the metareasoning overhead. By using the fastest alternative among the reasoning processes and not to have a concurrent metareasoning supervisor process and eliminating the post-metareasoning, we have the fast reasoning exactly as fast as without metareasoning.

4.1 Pre-metareasoning

The meta-level reasoning will select the best choice among the paths before running them. The selection is based upon encoded background knowledge of expert in most cases. When (1) we have not enough expertise for selecting the reasoning path, (2) the input data is in- complete to do path-selection metareasoning, and (3) con icted constraints do not converge to a unique answer, the pre-metareasoning fails.

All the metareasoning processes which deal with utility estimation [41], the contract any- time algorithms [44] and any other metareasoning system which does its job by prediction and selection of the best deliberation path, does pre-metareasoning.

All the systems that plan for reasoning, are in this category. Most of the definitions of metareasoning in early years, address only this category of metareasoning. So, we see that the proposed classification is covering some new approaches in which start the reasoning and control it in parallel with reasoning.

4.2 Para-metareasoning

In systems that metareasoning runs as a separate task and does handshaking steadily or in some inter-process communication points, the system will do para-metareasoning. One possible architecture to implement the para-metareasoning is a system with interruptible anytime algorithms [44]. It is obvious that the architecture of an agent powered by parametareasoning, should be di erent. The agent should have hierarchical concurrent communicating processes for reasoning and metareasoning. Lacking such an architecture prevents us to embed para-metareasoning within the agent.

Although this class of metareasoning is already implemented in some cases, there has been no precise unification among those cases to put them in a single category.

4.3 Post-metareasoning

In the situations that we have time to run a good amount of equivalent processes and postpone the selection of them, we have the chance to select the returning values of those processes instead of selecting the processes themselves. To have safe environment and not let one process to have effect on the data required by others, these processes should not use shared writable data to be isolated enough. They make good anytime algorithms. The more time they have, the more processes they can run. So they have more chance to find a better answer.

Post-metareasoning is ideal when we have too much candidate (alternative) processes for a single task. Especially, when the evaluation function (which selects the best process) takes more running time compared with any of the processes. So we can run one (preferably fast) process to have an answer and then try to run others which require more time.

Post-metareasoning is useful in cases that we can not pre-determine the best algorithm by our knowledge. And there is no utility estimation function at all. In these cases there could be some functions which work on the returned values for minimizing the errors. In the next section we will see one of these cases. It will be shown that post-metareasoning will help us choose among results of the different self-localization algorithms. This type of metareasoning has not been mentioned in any references up to the best of the knowledge of the authors. The importance of this category is mostly in its power to generate knowledge for the pre-metareasoning on the y. It means that, when we can not pre-determine the best reasoning path, we use the post-metareasoning to sketch the best and after a while, we have the chance to know the best before starting the reasoning according to statistical (or other) outcome. In cases that there is not an always the best path, at least we know that we should continue the post-metareasoning.

5. Post-metareasoning in a Case Study

Many computational solutions for real world problems could not give acceptable outputs in special circumstances. One of these problems is self-localization which deals with estimating the pose (position and orientation) of an agent. There are three most frequently used self-localization algorithms. Grid-based Markov localization [6, 14] could not be exact enough when computational considerations force us to make the grids with low resolution. But it works fine in global search of the agent position. Kalman Filtering method for self-localization [19, 22, 25, 28] has a better precise output but it is unable to globally localize the agent or to recover from total localization failures [18]. The Monte Carlo method of localization [12, 13, 24] is really ine cient when an external system changes the location of the agent [18].

5.1 Better self-localization

To have a better self-localization, the combined and mixed methods have been proposed and evaluated [18]. The alternative approach is to choose the best self-localization method on the fly. So, instead of mixing the methods, all methods run apart and isolated from each otherand only an

external evaluation function selects the winner. This will keep us safe from the situations in which the statistically great algorithms fail repeatedly in some time intervals. The finest paradigm is to select one algorithm according to the situation and predict the best beforehand by pre-metareasoning. So we will use just one computational path, among the alternatives and reduce the required processing power. But when the (1) situation analysis takes a long time or (2) situation analysis fails to predict the best algorithm, or (3) we need to know how perfect the prediction was done in tests, we do our algorithm selection after running all our algorithms in a way called here as post-metareasoning.

To make a testing environment for this approach, we need more than three algorithms to have more choices. Voting with only two people is not that wise. These algorithms should also have almost equal precision and we should not be able to select one of them as "always the best"[a]. So it is not possible to mix two less powerful methods to above three great ones. We have to find five simple trigonometric self-localization methods and three most frequently used methods (see above) are not used. The goal is just to determine the effect of a post-metareasoning algorithm selection system and not the validity, correctness or efficiency of each individual algorithm. This approach is neither an e ort to find a new self-localization algorithm nor to prove empirically the usefulness of post-metareasoning. We use this example just to explain post-metareasoning better.

5.2 Software robot

The software robot (softbot) used as the test bed, is a soccer playing robot. It is an agent which its sensors and actuators are connected over UDP/IP protocol. The softbot as defined in RoboCup simulation league, has been chosen and designed [39, 38] as a base. Then it has been re-implemented in Lisp for the availability of tools. At startup, the Lisp interpreter loads the virtual player (softbot) into memory. Next, it evaluates the elements of the positioning KB which contains five positioning routines. It repeatedly modifies the correctness values of elements in KB dealing with positioning. In other words, the self-localization procedures have varying levels of correctness. These levels of correctness have been set according to the positions calculated by all procedures. *The closer two positions we have, the more scores we give to their originating procedures.*

Consider that we keep all alternative procedures in a list called Alternatives. As figure 1 shows. there are four lists of alternative procedures: X and A and B and C .Each corresponding list has the name of the main function (say X), its evaluation function (say *EvalX), and a number of function names with their correctness values. Our goal is to use* the best function/returning value when the main function is called.

```
setf Alternatives (
   (X EvalX ( X01 7 ) ( X02 15 ) ( X03 0 ))
   (A EvalA ( A01 2 ) ( A02 3 ))
   (B EvalB ( B01 4 ) ( B02 3 ) ( B03 3 ) ( B04 4 ) ( B05 4 ))
   (C EvalC ( C01 2 ) ( C02 1 ) ( C03 0 ) ( C04 9 ) ( C05 0 )  ( C06 0 )) )
(defun X02 (p1 p2) ( ... (A01 ...) (B05 ...) (C04 ...) ...))
(defun X01 (p1 p2) ( ... (A01 ...) (B05 ...) (C06 ...) ...))
```

Figur1. Alternatives to show the correctness calculations

Method	X	Y	Point
Calcpos01	0.5	5.0	1
Calcpos02	1.0	1.0	2
Calcpos03	4.0	3.0	3
Calcpos04	2.5	3.5	4
Calcpos05	3.0	1.5	5
Average	2.2	2.8	

Figure 2. The (x, y) values returned by five self-positioning procedures at one time

Figure 2 shows the sample concurrent output of all five self-positioning procedures at a specific random time. In this figure (2) the Average point has been shown. This point is calculated in our tests to measure how the output of proposed method is compared with the average. Figure 3 shows the positions of calculated points relative to X and Y axis.

Any self-positioning method generates a position which could be selected as the best one. The "Best" position is one of the positions which is near the others. To be precise, we assign a higher score to both end points of the shortest line interconnecting positions. The more distance between two end points, the less score will be assigned to both end points. Any end point will get four scores (n -1scoresforn alternative coordinates). By adding the scores of a coordinate to each other, we have the total score. The coordinate with greatest total score will be selected.

The third point in figure 4 has four scores: 0.8, 0.7, 0.3, 0.2 from top to the bottom. After adding all scores in each column, the best point (number four in this example) will be chosen. In practice we assign the scores from the bottom to the top. The last row (at the bottom) in the figure 4 will always get 0.1scores. Sothed1andd5 both will have 0.1 scores. The next row (from the bottom) has less value in its *Length column. This means that we add 0.1 to the score of this row (d1 and d3 will* have 0.2 scores). The row of d12 will have the same score, because its Length field has not been reduced compared by its lower row. By repeating the above steps for all rows, the table will be filled as shown in figure 4.

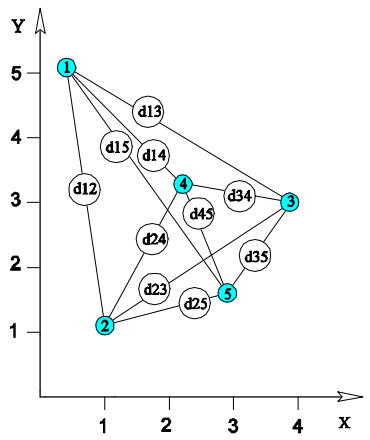


Figure 3. The schematic view of five concurrent outputs generated by self-positioning procedures

Line	Length	1	2	3	4	5
d34	1.5811	-	-	0.8	0.8	-
d35	1.8027	-	-	0.7	-	0.7
d25	2.0615	-	0.6	-	-	0.6
d45	2.0615	-	-	-	0.6	0.6
d14	2.5	0.5	-	-	0.5	-
d24	2.9154	-	0.4	-	0.4	-
d23	3.6055	-	0.3	0.3	-	-
d12	4.0311	0.2	0.2	-	-	-
d13	4.0311	0.2	-	0.2	-	-
d15	4.3011	0.1	-	-	-	0.1
	Total:	1.0	1.5	2.0	2.3	2.0

Figure 4. The scores of each coordinate calculated with respect to their distance from others

The errors of the methods gathered based on the real and exact coordinates reported by the simulator. To check the results, we have changed the simulator and forced it to send the real position of player at the end of the SEE message[b]. After the change, the received message from the soccer server (simulator) became (SEE ... (RP 10 15)) that means the real position of player is **(10 15)** or **x =10** and **y = 15**.

5.3 Performance analysis

Figure 5 shows the error of six calculated values for the player's position. By error we mean the length of a line connecting the calculated position and the real position As we see in figure 5 the errors of di erent methods are between zero and two meters. We have assigned the value of 2 to the error in cases when the method fails due to lack of required object(s).

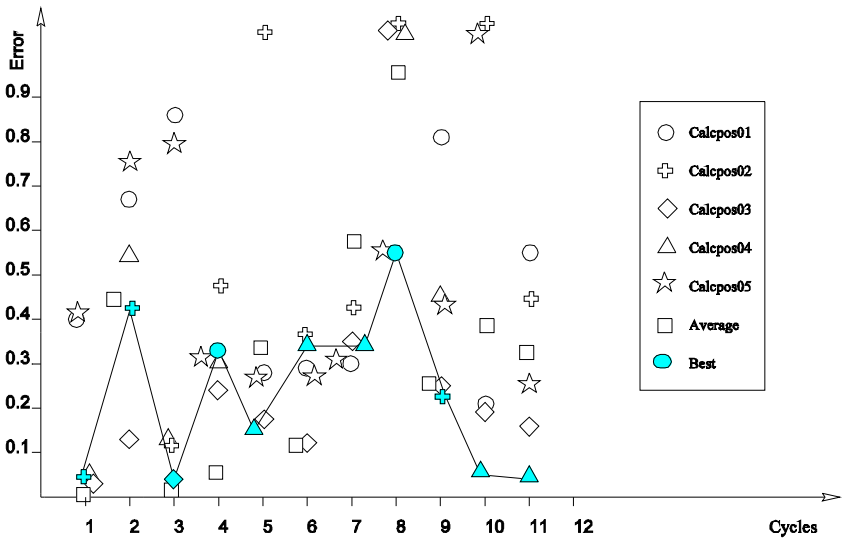
In all tests, the "Best" position is mostly one of the two coordinates with least linear errors (around 90% of observations). Besides, the bad coordinates attract the "Best" one toward themselves. Means that, the "Best" coordinate will seldom become the one with least linear error (only in 15% of situations).

Loosing the chance to be the really best one, is a disappointing consequence of having too bad coordinates. At first, we may imagine that this will make all our selections wrong, but how much wrong? There is another method to see total errors. To do so, one should notice that the good stand alone methods (like say CalcPos03) sometimes have really bad results too. And the better mechanism to compare them is to sketch the graph of accumulative linear errors of each method. This has been done and is illustrated in figure 6.

It is important that the "Best" position is not the point with average **x** and average **y** like the point shown in figure 2. To show this, we also calculated the coordinates of the average position. And in figure 5, the average point is not even near the "Best" selected position, which has been filled to be different.

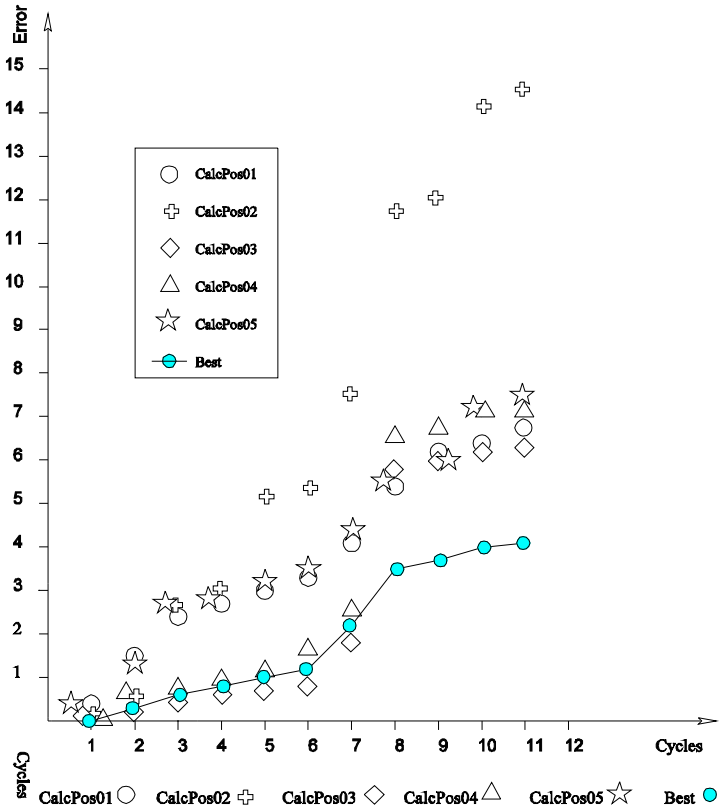
6. Concluding Remarks

Metareasoning, here has been under focus as a key in a complex and messy path to make programs that will find their own bug and will fix that bug themselves. Any program that changes its execution path by (say) choosing a different function each time, may change the critical parts, dealing with intelligent tasks.



Cycles	CalcPos01 ○	CalcPos02 ✚	CalcPos03 ◇	CalcPos04 △	CalcPos05 ☆	Average □	Best ●
1	0.40312	0.01926	0.01224	0.01927	0.40312	0.0	0.01926
2	0.67833	0.42149	0.13049	0.54220	0.77233	0.42427	0.42149
3	0.86017	0.10945	0.04225	0.13971	0.81261	0.02573	0.04225
4	0.33207	0.47471	0.24399	0.30862	0.31103	0.05806	0.33207
5	0.28844	2	0.17553	0.15123	0.28032	0.33480	0.15123
6	0.29079	0.36817	0.12764	0.36493	0.28876	0.12563	0.36493
7	0.30721	0.42479	0.35013	0.35013	0.31978	0.57168	0.35013
8	0.55548	2	2	2	0.55548	0.95559	0.55548
9	0.81001	0.22449	0.25551	0.45477	0.44528	0.25559	0.22449
10	0.21174	2	0.15318	0.07420	1.12010	0.38880	0.07420
11	0.55197	0.44703	0.16674	0.05879	0.25771	0.32340	0.05879

Figure 5. The linear distance of calculated positions and real position. As shown, the Best is the one which is filled



Cycles	CalcPos01 ○	CalcPos02 ✣	CalcPos03 ◇	CalcPos04 △	CalcPos05 ☆	Best ●
1	0.40312	0.01926	0.01224	0.01927	0.40312	0.01926
2	1.48457	0.46002	0.15498	0.58076	1.57857	0.46002
3	2.42379	2.56947	0.35685	0.82244	2.57003	0.58133
4	2.75586	3.04419	0.60084	1.13107	2.88107	0.91340
5	3.04430	5.04419	0.77638	1.28230	3.16139	1.06463
6	3.33510	5.41237	0.90403	1.64724	3.45015	1.42956
7	4.12574	7.83717	1.77994	2.51310	4.24484	2.30547
8	5.39177	11.83717	5.77994	6.51310	5.51087	3.57151
9	6.20179	12.06166	6.03546	6.96787	5.95616	3.79600
10	6.41354	14.06166	6.18864	7.04208	7.07626	3.87021
11	6.96551	14.50870	6.35539	7.10087	7.33397	3.92900

Figure 6. The accumulative error of each single method compared with the “Best” one

Metareasoning, is a tool to have the safe modification in the programs. By metareasoning approach we can start our way toward general intelligence and this can be done by making programs with creativity (see section 1 or [5]), learning the learning [40] and a teachable intelligent machine [32]. To do so, we have to make systems capable of handling all classes of metareasoning.

The proposed classification opens new areas of challenge for AI system architects who need to design highly flexible systems with long life time to have the better return of investment. So, we can be hopeful about more investment on this kind of AI powered systems. Such systems may be used in control of sensitive systems.

They are supposed to be a base for software systems that are able to take part in their own testing. So, the programmer should not remove the old version of his/her code when the new version of a function is ready. In contrast, if the function fails to satisfy the predetermined performance, the system itself reinserts the old version automatically.

There are many possible aspects to classify metareasoning. It is obvious that we may have metareasoning, before, during and after domain-level reasoning. This taxonomy is important as it can have a great influence in the design of new intelligent systems. By making a system with all three above types of metareasoning, we have a complete metareasoning *system. This system will have an steady and continuous metareasoning in parallel with low* level reasoning with at least three points of interaction between reasoner and meta-reasoner.

Although the primitive self-localization methods in this paper are not among the best and sophisticated ones, they are good enough to demonstrate the usefulness of post-metareasoning. These methods are simple enough to be calculated for several times during the simulation period in a not that fast (interpreting) Li sp environment. The methods could be replaced by sophisticated ones, to dynamically choose the superior pose determination algorithm on the fly. Such a try requires a good computing power (a fast processor) and an optimizing programming language. It is also recommended that, the very bad primitive methods with large linear errors are not added to primitive methods. Such methods will make the overall results unstable.

In figure 6 we see that the selected position with the proposed use of post-metareasoning does better after

repeated cycles. The proposed post-metareasoning method takes over very soon when some self-positioning methods lack the necessary input data or have an inaccurate output.

To have a better qualification for the proposed "Best" point selection process, it is possible to generate the results of other simple or complicated heuristics and compare these all. Figure 7 compares the accumulative errors of some other methods. It is shown that based on the last scores in previous simulation cycle, we could not have a good prediction. In other words, the points labeled as "Last" in figure 7 are not doing that good. And this has a more important result: the selected method is always c hanging. Besides, even if we use scores of methods in two recent cycles, and select the output of a method which has been the best among them, the result becomes only a little bit better. The points labeled as "2-Last" in figure 7 show the matter.

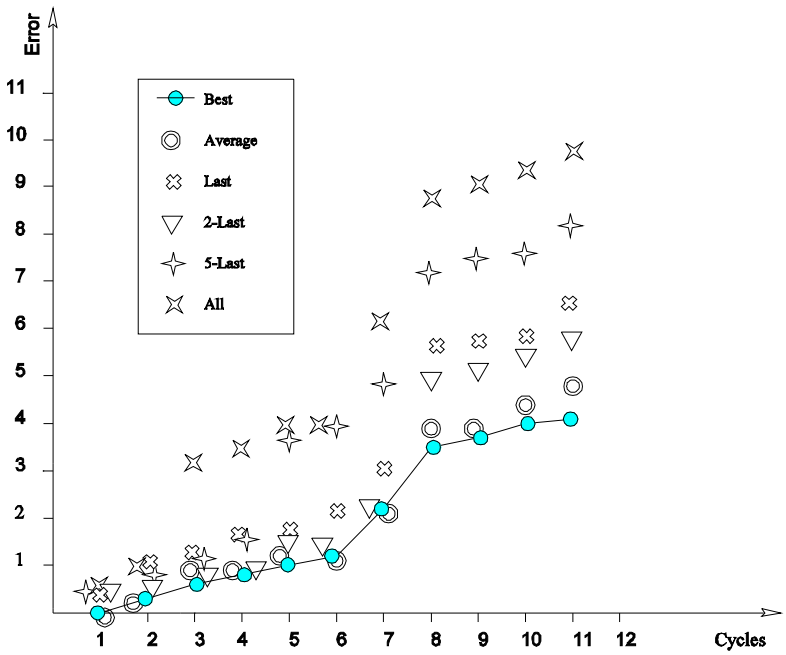
To better evaluate the methods which are based on history, we also have the "5-Last" and "All" points, which are the points generated by methods which were better in last five cycles and from the beginning respectively. These also did not produce better results.

This paper is not to show a specific self-localization method. Instead it is to show the use of meta-reasoning in spatial inference. The evaluation function, chooses one of the localization procedures. It is to infer what reasoning procedure is the one with the best output.

The tests run for two weeks. Each daily run, produced more than ten tables similar to the tables at the bottom of figures 5, 6 and 7. All the tests, showed that, the positions found by the proposed post-metareasoning, have less errors than any individual positioning procedure. It is also better than the average position of all outputs generated by all procedures.

To check the possibility of reproducing such an output by history based fitness estimation, we tried to have a prediction for best procedure in next required localization. We tried last best procedure, the best procedure among last two invocations, the best one in last five invocations, and the best among all invocations so far. It was interesting that none of these estimation/prediction techniques, has better results compared with what post-metareasoning gives.

As we could have alternatives for the evaluation function, it is possible to try other evaluation functions in parallel with evaluation function and it is not required to check these functions in advance. We have never checked the self-localization functions in advance. It means that if the system itself learns a new method for self-localization or even the evaluation function(s), the best one among them may be selected and used later automatically by post-metareasoning.



Cycles	Best	Average	Last	2-Last	5-Last	All
1	0.01926	0.0	0.40312	0.40312	0.40312	0.40312
2	0.46002	0.42427	1.10071	0.55289	0.83685	0.96460
3	0.58133	0.88719	1.31214	0.74140	1.13617	3.00686
4	0.91340	0.94526	1.64421	0.98540	1.46824	3.31548
5	1.06463	1.28006	1.79544	1.13663	3.46824	3.60392
6	1.42956	1.40570	2.08623	1.42743	3.75904	3.73157
7	2.30547	2.21970	3.03681	2.21806	4.69957	6.08170
8	3.57151	3.64955	5.59230	4.77355	7.25505	8.79225
9	3.79600	3.90514	5.81679	4.99804	7.47955	9.04777
10	3.87021	4.29395	5.89100	5.15123	7.63273	9.25952
11	3.92900	4.61736	6.44297	5.70320	8.18471	9.70656

Figure 7. The accumulative error of history based methods and average of coordinates compared with the "Best" one

References

[1] J. F. Allen. "AI Growing up the Changes and Opportunities," *AI Magazine*, vol. 19, pp. 13-23, 1998.

[2] G. Attardi and M. Simi, "Meta-level Reasoning Across Viewpoints," In T. O'Shea, editor, European Conference on Artificial Intelligence, pp. 315–325, Amsterdam, North-Holland, 1984.

[3] K. A. Bowen and R. A. Kowalski. "Amalgamating Language and Metalanguage in Logic Programming,". In K.L. Clark and S.A. Tarnlund, editors, Logic Programming, pp. 153–172. Academic Press, London, 1982.

[4] R. J. Brachman, "The Future of Knowledge Representation," *In AAAI-90*, 1990.

[5] B. Buchanan, "Creativity at the Metalevel," *AI Magazin*e, vol. 22, no. 3, pp.13–28, 2001.

[6] W. Burgard, D. Fox, D. Hennig and T. Schmidt, "Estimating the Absolute Position of a Mobile Robot Using Position Probability Grids," *In 14th National Conference on Artificial Intelligence (AAAI'96),* pp. 896–901, 1996.

[7] V. Conitzer and T. Sandholm, "Definition and Complexity of Some Basic Metareasoning Problems," *In International Joint Conference on Artificial Intelligence (IJCAI)*, 2003.

[8] K. S. Cook and M. Levi. The Limits of Rationality University of Chigago Press, 1990.

[9] S. Costantini, "Semantics of a Metalogic Programming Language," *Intl. Journal of Foundation of Computer Science,*no. 1, 1990.

[10] S. Costantini, Meta-reasoning: a survey. In A.C. Kakas and F. Sadri, editors, *Lecture Notes in Computer Science, Computational Logic: Logic Programming and Beyond, page 253. Springer-Verlag Heidelberg, Dipartimento di Informatica Univer*sitra degli Studi di L'Aquila, via Vetoio Loc. Coppito, I-67100 L'Aquila, Italy stef-cost@univaq.it, 2002.

[11] T. Dean and M. Boddy, "An Analysis of Time-Dependent Planning," *In Proceedings of Seventh National Conference on Artificial Intelligence,* pp. *49–54,* Minneapolis ,Minnesota, 1988.

[12] F. Dellaert, D. Fox, W. Burgard and S. Thrun, "Monte Carlo Localization for Mobile Robots," *In IEEE International Conference on Robotics and Automation (ICRA99),* 1999.

[13] D. Fox, W. Burgard, F. Dellaert and S. Thrun," Monte Carlo Localization: Efficient Position Estimation for Mobile Robots," *In Proc. National Conference on Artificial Intelligence (AAAI'99), 1999.*

[14] D. Fox, W. Burgard and S. Thrun, "Markov Localization for Mobile Robots in Dynamic Environments," *Journal of Artificial Intelligence Research*, no.11, 1999.

[15] F. Giunchiglia and A. Cimatti, Introspective metatheoretic reasoning. In L. Fribourg and F. Turini, editors, Logic Program Synthesis and Transformation - Meta-*Programming in Logic. LNCS 883, pages 425–439. Springer-Verlag, 1994.*

[16] Irving Good. Twenty-seven principles of rationality. In V. Godamble and D. Sprott, editors, Foundations of Statistical Inference, pages 108–141. Holt, Rinehart, Wilson, Toronto, 1971.

[17] B. Grosz and R. Davis," A Report to Arpa on Twenty-First Century Intelligent Systems," *AI Magazine,* vol. 15, no. 3, pp. 10-20, 1994.

[18] J.-S. Gutmann and D. Fox, "An Experimental Comparison of Localization Methods Continued," In *Proceedings o f the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'02),* Lausanne, Switzerland, 2002.

[19] J. S. Gutmann, T. Weigel and B. Nebel, "A Fast, Accurate, and Robust Method for Self Localization in Polygonal Environments Using Laser Range Finders. *Advanced Robotics Journal,* vol. 14, no. 8, pp. 651-668, *2001.*

[20] B. Hayes-Roth, "An Architecture for Adaptive Intelligent Systems, " *Articial Intelligence*, vol. 72, pp. 329-365, 1995.

[21] M. Hearst and H. Hirsh, "AI's greatest Trends and Controversies," *IEEE Intelligent Systems, 2000.*

[22] L. Iocchi, D. Mastrantuono and D. Nardi," A Probabilistic Approach to Hough Localization," *In International Conference on Robotics and Automation (ICRA),* Seoul, Korea, 2001.

[23] D. Lenat, R. V. Guha, K. Pittman, D. Pratt, and M. Shepherd, "Cyc: Toward programs with common sense," *Communications of the ACM*, vol. 33, no. 8, pp. 30-49, 1990.

[24] S. Lenser and M. Veloso," Sensor Resetting Localization for Poorly Modeled Mobile Robots," *In Int. Conf. on Robotics and Automation (ICRA),* 2000.

[25] J. Leonard and H. Durrant-Whyte, "Mobile Robot Localization by Tracking geometric beacons," *IEEE Transaction on Robotics and Automation*, vol. 7, no. 3, pp. 376-382, 1991.

[26] J. Lighthill, Artificial Intelligence: A General Survey, 1973.

[27] J. Malenfant, G. Lapalme and G. Vaucher, "Objvprolog: Metaclasses in logic," *In ECOOP'89*, pp. 257–269. Cambridge Univ. Press, 1990.

[28] P. S. Maybeck, "The Kalman Filter: An Introduction to Concepts, "In Cox and G. Wilfong, editors, Autonomous Robot Vehicles, pages 194–204. Springer-Verlag, 1990.

[29] J. McCarthy, M. L. Minsky, N. Rochester and C. E. Shannon. A Proposal for the Dartmouth Summer Research Project on Artificial Intelligence, 1955.

[30] J. McCarthy, "Review of Artificial Intelligence: A General Survey, "*Artificial Intelligence,* vol. 3, no. 3, 1974.

[31] T. Menzies, "21st-Century AI: Proud, not Smug," *IEEE Intel ligent Systems*, 2003.

[32] D. Michie. The very early days, October 2002.

[33] T. M. Mitchell et al, Theo: A Framework for Self-improving Systems. In K. VanLehn, editor, Architectures for Intelligence. Erlbaum, 1989.

[34] A. Newell and H. A. Simon, "Computer Science as Empirical Inquiry: Symbols and Search," *Communications of the ACM*, vol. 19, pp. 113–126, 1976.

[35] R. M. Newman and E. I. Gaura, "Grand Challenges in Computing for the Twenty-First Century," *In Proceedings of A ISB Workshop on Grand Challenges for Computing Research.* UK Computing Research Committee, National e-Science Centre in Edinburgh, 2002.

[36] N. J. Nilsson, "Eye on the Prize," *AI Magazine*, vol. 16, no. 2, pp. 9-17, 1995.

[37] F. Pfenning, "Elf: A Language for Logic Definition and Verified Meta-Programming," *In Fourth Annual Symposium on Logic in Computer Science*, pp. 313–322, Pacific Grove, California. IEEE Computer Society Press, 1989

[38] S. Pourazin, "A Portable Architecture for Robocup Agents," In Proceedings of Agents 2000 *Conference,* Barcelona, Catalonia, SPAIN, 2000.

[39] S. Pourazin, "Simulated Soccer Robot Architecture: Toward Being on Time," *In 18th IASTED International Conference on Applied Informatics (AI'2000), Innsbruck, Aus*tria, 2000.

[40] S. Pourazin and A. A. Barforoush, "Fleng The Flexible Inference Engine," *In Proceedings of 18th IASTED Int'l Conference on Applied Informatics (AI'2000), Innsbruck,* Austria, 2000.

[41] S. Russel and E.Wefald, "Principles of Metareasoning," *Artificial Intelligence*, vol. 49, 1991.

[42] S. J. Russell. Metareasoning. In Robert A. Wilson and Frank Keil, editors, MIT *Encyclopedia of the Cognitive Sciences. MIT Press, 1998. Parts available from:* http://cognet.mit.edu/MITECS/Entry/russell.html.

[43] S. J. Russell and E. H. Wefald. Principles of metareasoning. In R. J. Brachman, editor, *Proceedings of first International Conference o n Principles of Knowledge Representation and Reasoning, San Mateo, California. Morgan Kaufmann,* 1989.

[44] S. J. Russell and S. Zilberstein, "Composing Real-Time systems," *In Proceedings of the Twelfth International Joint Conference on Artificial Intelligence*, pp. 212–217, Sydney, Australia, 1991.

[45] S. C. Shapiro. Encyclopedia of Artificial Intelligence. John Wiley and Sons, Inc., 1990.

[46] M. R. Shinwell, A. M. Pitts and M. J. Gabbay. "FreshML: Programming with binders made simple," *In 8th ACM SIGPLAN International Conference on Functional Programming (ICFP 2003), Uppsala,* pp. 263–274. ACM Press, Sweden, 2003.

[47] H. A. Simon. Models of Bounded Rationality, vol. 2, MIT Press, 1982.

[48] A. Sloman, AI and the Study of Mind, October 2002.

[49] B. C. Smith. Reflection and Semantics in Lisp. Technical report, Xerox PARC ISL-5, Palo Alto, CA, 1984.

[50] E. Stroulia and A. K. Goel, "Functional Representation and Reasoning in Reflective Systems," *Journal of Applied Intelligence*, vol. 9, pp. 101–124, 1995.

[51] A. M. Turing, Computing Machinery and Intelligence, *Mind*, vol. 59, pp. 433–460, 1950.

[52] T. Walsh, "Empirical Methods in AI," *AI Magazine*, vol. 19, no. 2, pp.13–28, 1998.

[53] D. Waltz, "Artificial Intelligence: An Assessment of the State-of-the-art and Recommendation for Future Directions," *AI Magazine*, vol. 4, pp. 55–67, 1983.

[54] R. W. Weyhrauch, "Prolegomena to a Theory of Mechanized Formal Reasoning," *Artificial Intelligence*, pp. 133-70, 1980.

---

[1] Otherwise, we can use the "always the best" and forget the other choices.

[2] The normal simulator only sends the distance and orientation of objects in its eyeshot relative to the robot, similar to what a real sensor reports.

Shahriar Pourazin received the B. Sc. degree on Software Engineering in 1990 from Sharif University of Technology, Tehran IRAN. He received the M. Sc. degree on Software Engineering in 1994 from Sharif University of Technology and is now a Ph. D. candidate in Computer Engineering Dept., Amirkabir University of Technology, Tehran, Iran.

Ahmad abdollahzadeh-Barforosh received the B.S. degree in accounting from Tehran University in Tehran, Iran in 1975. He received the M.S degrees in computer science from West Coast University, Los Angeles, california, U.S.A in 1980. He also received the Ph.D. in computer science from university of Bristol, England,1990. Dr. Abdollahzadeh served as visiting professor in department of computer science of Maryland university at college park, USA and Orsay university in Paris, France from 2000-2002. Currently, he is a nassociate professor in the IT & Computer Engineering Departement at Amirkabir University of Technology in Tehran, Iran. His research interests include, Information Retrieval, Artificiel Intelligence Techniques, Expert System, Natural Language Proceesing, Decision Support System, Knowledge Representation, Software Engineering.
Email: ahmad@ce.aut.ac.ir