International Journal of Information, Control and Computer Sciences
ISSN: 2517-9942
Vol:1, No:6, 2007

Decision Maturity Framework: Introducing Maturity In Heuristic Search

Ayed Salman, Fawaz Al-Anzi, Aseel Al-Minayes

***Abstract*—** Heuristics-based search methodologies normally work on searching a problem space of possible solutions toward finding a "satisfactory" solution based on "hints" estimated from the problem-specific knowledge. Research communities use different types of methodologies. Unfortunately, most of the times, these hints are immature and can lead toward hindering these methodologies by a premature convergence. This is due to a decrease of diversity in search space that leads to a total implosion and ultimately fitness stagnation of the population. In this paper, a novel Decision Maturity framework (DMF) is introduced as a solution to this problem. The framework simply improves the decision on the direction of the search by materializing hints enough before using them. Ideas from this framework are injected into the particle swarm optimization methodology. Results were obtained under both static and dynamic environment. The results show that decision maturity prevents premature converges to a high degree.

***Keywords*—** Heuristic Search, hints, Particle Swarm Optimization, Decision Maturity Framework.

I. Introduction

Heuristic-based optimization methods use heuristics clues (or hints) to advance the search process towards areas (in the search space) of potentially good solutions. Spanning the heuristic algorithms, we can deduce that heuristics are intelligent hints extracted from problem-specific knowledge used to direct the search toward areas favored at the time when they are obtained. Hints are usually extracted with respect to the current status of the search process. Thus, they are most often "local" and "immature": local in their affects and immature in their influence to the decision making process in the search algorithm. Utilizing such immature hints, optimization methodologies tend to loose their ability to search globally and therefore converge prematurely. Hence, an apparent solution for such problem is by controlling the hint utilization process and formulating it toward becoming more "mature"; mature in a sense that hints utilized only when they are mature enough to provide better (i.e., global) insight about the best direction to forward the search.

Decision maturity framework is can be used as a general way of enhancing the guidance system of the heuristic-based optimization methodologies. The framework is based on the very basic idea of accumulating individual hints (from a time window) and delaying their utilization until the end of the time window set by the algorithm. In this context, hints accumulation works as an incubator of their affects in order to provide an overall better affect on the decision than applying them individually. Philosophically speaking, one may think about the framework as a wiser brain versus an intelligent one: wiser brain tend to accumulate knowledge and experiences through time and build up a wisdom of multiple experience and information (i.e., many hints) that it can use for better decisions afterward. On the other hand, an intelligent brain tends to think intelligently under certain circumstances and provides an intelligent decision using available information (i.e., local hints) it can gather at that moment.

In this paper, we tested the validity of this framework by injecting some of the ideas into particle swarm optimization technique, and testing its suitability under problem with dynamic environment.

In the next section the decision maturity concept is described and the logic behind it is explained. Section III shows how this framework is applied to particle swarm optimization. Section IV presents experimental setting used to test the validity of the idea. Experimental results are shown in Section V. Section VI shows results of applying the idea under dynamic environments. Finally, Section VII presents conclusion of this paper.

II. Decision Maturity Framework

To illustrate the decision maturity framework we preferred using a life example to make the idea more easier to grasp. Imagine the situation where a frog and turtle are having a race to who reaches a piece of food first. We assume that there is only one piece of food in the forest (and both animals have an appetite for). Both animals search the forest for the food by continuously moving in directions that each think the food is located. The two animals are smart enough to learn from their own movement. There is one major difference between the two animals that is the step size. While the turtle tend to take small steps the frog has the tendency to take long leaps. Because the frog has to use more energy in leaping, it has to rest for a while before it can make the leap. Meanwhile, the turtle continues to move during that resting period of the frog and manages to walk a distance equivalent to that the frog has made in its last leap. Given this scenario, the question

Manuscript received January 16, 2004.

Ayed Salman, , Fawaz Al-anzi, Aseel, Computer Engineering Department, Kuwait University (Tel: 965-4985833 KUWAIT; e-mail: {ayed/alanzif}@eng.kuniv.edu.kw)

Aseel Al-Menyes, Master Student, Kuwait University, {e-mail: aseel@eng.kuniv.edu.kw).

remains; who will reach the food first?

Since both animals are trying to explore more ground to find food, it is logical that both animals will be probably making almost equal number of moves in wrong directions from the food to discover the right general direction for the food. Hence, it is logical to assume that the turtle will be getting more smarter as time progresses since it make more moves the frog is getting its rest. But because it is slower, there is no grantee that it will reach the food before the frog DOES. A wise frog comes into the forest that will behave differently from our previously described frog. As the wise frog rest for its next leap, it will watch the movements of the turtle as it progresses and get a general idea of the knowledge it acquires due its continuous movement. The wise frog then will use this knowledge to enhance the direction of where it should leap as the resting period come to an end. This wise frog is most probably going to find the food faster than the turtle because it is faster and getting as smart as the turtles, so it will be always few steps ahead of the turtle in THE RIGHT DIRECTION AS IT LEAPS.

To illustrate the picture mathematically, we will use a simplified version of it. We will assume that the turtle can make a step of length x every minute and the frog makes a leap of 2x every two MINUTES AS SHOWN IN FIGURE I. At start, the two animals are in the same place. The food is at distance L with and angle t0 from both. The turtle will make a first step of length x in angle t1. The next turtle step is a step of length x in angle t2 from the position it stopped at form the first step. Because the wise frog is watching the movement of the turtle, when the time comes for a leap it will choose to leap its 2x step on same path as the direction of the last step of the turtle, with an angle t3. This smart move of the wise frog will ensure that if the food is in the current path of the turtle, the frog is also getting closer to it, and most of the time faster than the turtle. It is not difficult to mathematically prove that, if the food is still far enough, at the end of 2 minutes the frog is always closer to the food than the turtle using this strategy. However, as the two animals get closer to the food, smaller step strategy tends to be more logical to use. This strategy of the wise frog has been proven to produce more maturity on decisions of movement.

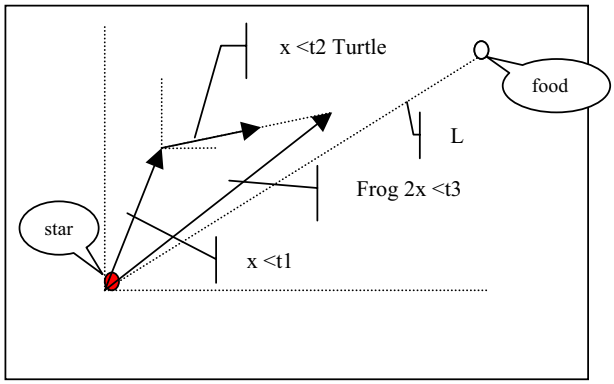

Figure I: Graphical representation explaining the logic behind Decision maturity framework

III. PARTICLE SWARM OPTIMIZATION (PSO) WITH DECISION MATURITY

We introduce Particle swarm optimization in the following subsection and then we show how its can be modified within the decision maturity framework.

A. 3.1 Particle Swarm Optimization

Particle swarm optimization (PSO) is a population-based evolutionary computation technique developed by Dr. Eberhart and Dr. Kennedy in 1995 [4]. It was inspired by social behavior of bird flocking or fish schooling. The system works by initializing a population with random solutions to the problem at hand and searching for optima through creation of multiple generations. In PSO, the potential solutions called particles and are "flown" through the problem space by following the current best particle behaviors.

Each particle keeps track of its coordinates in the search space. These coordinates associate to a potential solution in the problem domain. Particles are evaluated according to a fitness they achieved so far. The algorithm keeps track of two potential decision influencers: Best solution each particle achieve up to the moment of the decision (called lbest) , and best solution achieved by all particles (called gbest). These two attracters guide the search process toward areas of potentially good solutions. PSO uses a model that intermix the affect of both attracters. The particle swarm optimization idea relies on accelerating the velocity of each particle toward a location somewhere in the distance between its gbest and lbest locations. Acceleration factors are randomly weighted in order to avoid bias towards one attractor over the other. Researchers find particle swarm optimization an attractive technique because there are few parameters to adjust. PSO has been used across a wide range of applications with large success.

B. Simple PSO

Simple PSO [4] consists of swarm of particles moving in an n-dimensional search space where the fitness (i.e quality measure) can be calculated. Each particle has a position represented by a position-vector X and a velocity represented by a velocity-vector V. Each particle remembers its own best position so far in a vector lbest, which represents the position where it achieved its best fitness. In addition, a neighborhood relation is defined for the swarm. The best position vector among all the neighbors of a particle is then stored in the particle as a vector gbest. Velocities are updated every time step then particles moves to a new position according to the new velocities. Velocities are updated with the following equation:

$$V_{new\,i} = W * V_{old\,i} + rand\,() * (lbest - X_i) \quad (1)$$
$$\ldots\ldots + Rand\,() * (gbest - X_i)$$

Then, the new position is determined with the following equation

$$X\_new_i = X\_old_i + V\_new_i \quad (2)$$

Where *V_new* is the new velocity vector of particle,
X is the particle position vector,
rand() and *Rand()* are both real numbers generated randomly between (0,1) to remove bias towards any of the two (global and local) attractors.
The parameter *W* (inertia weight) is a real number that controls the affects of magnitude of the old velocity towards calculating the new velocity.

The Simple PSO algorithm is summarized as the following:

- Loop until meeting termination criteria
- Calculate the fitness of all particles.
- Record the best experience so far for every single particle (i.e. best position visited so far: *lbest*).
- Record the global leader position among different particles (*gbest*).
- For each particle *I*, for each dimension *d*; Adjust velocity and position for each particle according to the following equation:
- $$Vnew_{id} = W * Vold_{id} + rand() * (lbest - X_{id})$$
  $$\ldots\ldots + Rand() * (gbest - X_{id})$$

$$X\_new_{id} = X\_old_{id} + V\_new_{id}$$

C. Injecting Decision Maturity Into Particle Swarm Optimization

The decision maturity framework was injected into the step of updating position and velocity of particles in the simple PSO algorithm. First, PSO attractors (i.e. *lbest* and *gbest*) effects are separated rather than mixed in equations used by simple PSO. Two factors are being accumulated separately: *gbest* velocity factor and *lbest* velocity factor. We dealt with these factors as separate hints. We changed the algorithm such that those hints are accumulated during a randomly varying period of time, then applied to the PSO process at the end of this period. Accumulation is repeated again for another random period of time, and so on. The basic algorithm is summarized as following:

- Loop until termination:
- Calculate the fitness of all particles.
- Record the best experience so far for every single particle (i.e. best position visited so far p_Local).
- Record the global leader position among different particles (p_global).
- For each particle *i*, for each dimension *d*; Adjust velocity and position for each particle:
- Initialize two random numbers *r1* and *r2*.
- If *r1* > *r2*, accumulates hints (don't apply them yet):
  $$\text{hints}_{id} = \text{hints}_{id} + rand() * (lbest_{id} + X_{id})$$
- Otherwise, apply accumulated hints
  $$\text{hints}_{id} = \text{hints}_{id} + rand() * (gbest_{id} + X_{id})$$
  $$V_{id} = W * V_{id} + \text{hints}_{id}$$
  $$X_{id} = X_{id} + V_{id}$$

TABLE I: DE JONG FUNCTIONS

Name	Expression	Range
Rastrigin	F(x)=∑{xi^2–10cos(2Πxi) +10}	-5.12 ≤x≤ 5.12
Sphere	F(x)= ∑ xi^2	-5.12 ≤x≤ 5.12
Griewank	F(x)= 1/4000 ∑ xi^2 –∏ cos(xi/√i) +1	-600≤ x ≤ 600
Schewefel	F(x)=418.9829-xi sin(√ (abs(xi)))	-500 ≤x ≤500

IV. Experimental Settings

Four hard optimization functions developed by De Jong [4] used in the experiments as shown in Table I. These functions were used to compare simple PSO to PSO with maturity framework embedded. These functions are widely known and used (by many researchers) as benchmark functions for optimization techniques such as Genetic Algorithm and PSO.

Both the simple PSO performance (measured in fitness yielded) and our modified algorithm were compared thoroughly using variation of different parameters such as iteration number and problem dimension's size parameters. The *diversity* (i.e. the measure of differences between location of particles in the population) and the fitness of each method are recorded.

The diversity function is defined as:

$$Diversity = \frac{\sum_{i=1}^{S} \sqrt{\sum_{j=1}^{N} (p_{ij} - p_j)^2}}{S} \quad (3)$$

$$F = \sqrt{\sum_{i=1}^{D} (g_i - x_i)^2} \qquad (4)$$

Where S is the population size, N is the dimensionality of the problem, pij is the jth value of the ith particle and p_j is the jth value of the average point p.

V. Experimental Results

We devised two variation of the same idea of embedding the maturity into PSO. The difference between the two is the "how-about" of application of hints: method1 applies hints as shown in the algorithm described in section II;

Whereas method 2 applies hints in a deterministic order (rather using randomization to determine the timing of application) but

very similar to the mechanism above. Results shown in Table II shows that both methods utilizing the maturity idea beat the simple PSO in both measures: fitness and diversity of best solutions obtained by any algorithm. We see that embedding the idea leads towards better fitness functions and slower convergence (i.e. larger diversity).

VI. Decision Maturity In Dynamic Environments

In previous sections, the framework of Decision Maturity idea was tested under static problem search spaces (e.g. De-jong problems). In this section, we extend the experimental analysis into dynamic environments. Many real-world problems has a dynamic nature where objective changes through time and/or space. The main objective of this type of analysis is to explore the strength of the proposed framework when it encounters problem with such nature. In particular, we are interested in two criteria: performance of the method, compared to others, in achieving a good solution and maintaining more diverse population through time. The PSO algorithm with embedded decision maturity idea implemented for static environment in previous sections was adapted to handle the dynamic environments.

The dynamic fitness function depicted in equation measure the distance of the particle xi from the goal gi, which moves at some random velocity through the search space D. Particles attempt to locate the goal on the basis of the strength of the goal signal. Equation 1 was used in [1].

In these experiments, we compared the three methods we described in previous section using the same settings as previous experiments and varying only the following: maximum allowed velocity of the particles, and number of generations to change the goal (i.e degree of goal dynamics). Table III and IV summarize the measurement results showing the average diversity each method attained during different iterations and the average best fitness obtained by each method. Results show clearly that the fitness of both suggested method within the decision maturity framework is much better than the simple PSO. This superiority gets clearer as we increase the number of generation to change the objective of the problem (i.e., the goal position). In addition, the diversity attained by methods incorporating decision maturity idea is much better than the

others, which don't do that. Instead, using this idea, a population-based algorithm can live longer and converge slower thus yield better results in dynamic environments.

From all these results we concluded that the idea of Decision Maturity has proven its efficiency when used in dynamic environment.

VII. Conclusion

Decision maturity concept can be applied to many real life application and can be embedded into many heuristic algorithm. In this paper, we embedded it into PSO. It achieved better results and maintains larger diversity in the population. In both the static and dynamic environments, the modified PSO have proven its efficiency and it yielded better fitness and diversity than the simple PSO. Currently, the approach is being investigated from different directions: Theory behind its success being developed using hint theory and Dempster-Shafer theory [3] [7] [8], real life application is yielded, encouraging results and being fully tested, and finally different situations where the decision maturity can be embedded into

TABLE II: RESULTS COMPARING SIMPLE PSO AND PSO WITH DECISION MATURITY FRAMEWORK

Fitness	**Simple_PSO**	**Method 1**	**Method 2**
Rastrigin	82.69643	67.74201	56.39222
Sphere	1.35795	1.05539	2.37642
Griewank	0.06527	0.01842	0.00777
Schewefel	6658.45752	2877.62988	2748.18311
Diversity	**Simple_PSO**	**Method 1**	**Method 2**
Rastrigin	0.00011	0.21540	0.33377
Sphere	0.00006	0.00013	0.00031
Griewank	0.00019	0.19325	0.08236
Schewefel	128.00948	314.60673	0.22739

different heuristic algorithms such as Genetic Algorithms and Simulated Annealing is being explored.

TABLE III: DIVERSITY MEASURED AS EQUATION (1) FOR DIFFERENT PSO CONFIGURATIONS USING 1000 ITERATION AND VARYING BOTH MAXIMUM ALLOWABLE PARTICLE VELOCITY AND THE NUMBER OF GENERATIONS TO CHANGE THE GOAL

MaxV			0.5	1	5	50	500
Number of generations to change the goal.	1	PSO	0.0	0.0	0.0	0.0	0.0
		M1	429.6	445.5	426.7	444.6	442.0
		M2	295.6	300.3	299.7	301.0	271.9
	5	PSO	0.0	0.0	0.0	0.0	0.0
		M1	313.7	286.3	285.8	312.3	340.9
		M2	129.0	139.2	135.2	126.4	117.6
	50	PSO	8.0	703.3	9.4	12.1	8.6
		M1	227.6	1466.0	249.2	245.6	248.5
		M2	42.0	947.2	33.6	34.5	27.1
	100	PSO	11.8	10.5	11.3	12.4	11.4
		M1	181.5	166.1	164.0	197.9	182.7
		M2	20.3	20.6	19.9	20.3	23.4
	500	PSO	13.0	8.7	5.3	9.5	3.7
		M1	17.5	13.9	16.8	22.0	15.8
		M2	11.7	11.4	23.3	17.6	6.9

TABLE IV: FITNESS RESULTS FOR DIFFERENT PSO CONFIGURATIONS USING 1000 ITERATION AND VARYING BOTH MAXIMUM ALLOWABLE PARTICLES VELOCITY AND THE NUMBER OF GENERATIONS TO CHANGE THE GOAL.

MaxV			0.5	1	5	50	500
Number of generations to change the goal.	1	PSO	**4193.6**	**4141.1**	**4260.5**	**4154.4**	**4186.6**
		M1	**3936.3**	**3973.6**	**3930.3**	**4042.9**	**4008.6**
		M2	**3242.6**	**3302.1**	**3269.1**	3368.2	3929.8
	5	PSO	4202.4	4086.7	4165.7	4193.0	4230.0
		M1	3808.6	3880.6	3917.7	3875.1	3909.2
		M2	3533.1	3624.1	3706.2	3686.7	3905.2
	50	PSO	3134.4	2937.2	2814.2	2958.7	3066.8
		M1	2423.5	2116.0	2030.4	2208.9	2167.6
		M2	2629.2	2450.0	2415.0	2533.0	2719.8
	100	PSO	2175.1	1970.2	2004.7	2121.6	2284.3
		M1	1460.9	1390.0	1410.3	1447.2	1486.6
		M2	1872.4	1847.2	1868.3	1936.0	1892.2
	500	PSO	939.9	911.1	1020.0	897.3	1071.3
		M1	410.2	467.3	447.5	431.5	470.0
		M2	773.8	766.6	720.1	757.4	825.8

REFERENCES

[1] A. J Carlisle, (2000), Adapting PSO to Dynamic Environments, International Conference on Artificial Intelligence (ICAI 2000), Las Vegas ,.

[2] M. Clerc, (1999), The swarm and the queen: towards a deterministic and adaptive particle swarm optimization, Proceedings of the Congress on Evolutionary Computation (CEC99) (July) 1951-1957

[3] A. Dempster, (1968), Generalization of Bayesian inference. J. Roy. Statist. Soc. B., 30, 205–247.

[4] Y. Shi, R. Eberhart, (1999), Empirical study of particle swarm optimization, Proceedings of the Congress on Evolutionary Computation (CEC99) (July) 1945-1950.

[5] Jurgen Branke, (1999), Evolutionary Approaches to Dynamic Problems, A survey, Technical Report 387, Insitute AIFB, University of Karlsruhe

[6] Philippe Smets, (1996), Imperfect Information: Imprecision and Uncertainty. Uncertainty Management in Information Systems: pp225-254.

[7] Kohlas, J., & Monney, P. A. (1995). A mathematical theory of hints: An approach to dempster-shafer theory of evidence. Lecture Notes in Economics and Mathematical Systems No. 425. Springer- Verlag.

[8] Smets, P., & Kennes, R. (1994). The transferable belief model. Artificial Intelligence, 66, 191–234.