Applied Bionics and Biomechanics 9 (2012) 1–14
DOI 10.3233/ABB-2011-0033
IOS Press

Cognitive optimization in assistive living system development

Alan Bowling[a,*] and Fillia Makedon[b]
[a]*Department of Mechanical and Aerospace Engineering, University of Texas at Arlington, Arlington, TX, USA*
[b]*Department of Computer Science and Engineering, University of Texas at Arlington, Arlington, TX, USA*

Abstract. This paper presents an exploration of the characteristics and structure of a cognitive architecture for control of assisted living systems. The aspects of cognition considered are self-organization, communication, and inherited knowledge. A cognitive solution for a related problem, function optimization, is developed because of the complexity and size of the assistive living problem. Support for this approach stems from the artificial intelligence field where optimization is considered to be a critical aspect of cognition, and from the similarity between the performance metrics for the two problem domains. A search algorithm is developed using the bracketing and gradient methods as inherited knowledge. A key finding is that using a cognitive structure caused the search to display aspects of the characteristics, behavior, and performance of human cognition. In terms of performance, the cognitive search converges faster than either the bracketing or gradient searches alone, and its feasible problem set is larger than the intersection of their individual sets. Similarly, human cognition acts quickly and can address a large set of dissimilar problems. This gives confidence that the guidelines distilled from the development of the cognitive search can produce a similar level of performance when applied to an assistive living system. However, this paper does not address the details involved in actually implementing these guidelines on an assistive living system.

Keywords: Cognition, cognitive control, optimization, search, assistive living system

1. Introduction

This paper discusses an investigation into the development of some aspects of human cognition in the control of assistive living systems. The long term goal is to obtain a system that can quickly interpret large amounts of data taken from sensors located in the assistive living environment and determine some action in response. It is amazing to consider how quickly human attendants can perform this task using their cognitive abilities. The contribution in this paper is to illustrate how an exploration of cognitive function optimization can provide guidelines for the development of cognitive abilities in an assistive living system.

*Corresponding author. E-mail: bowling@uta.edu.

The complexities in cognition and assistive living environments make the development of cognitive control a daunting task. Thus the approach followed is to obtain insights from the development of a cognitive solution in a different problem domain, function optimization, with similar characteristics and performance metrics. Further justification for this choice stems from the artificial intelligence community in the suggestion that the ability to optimize is a key aspect of cognition [10, 15]. Accordingly, a search algorithm formulated using ideas from cognitive science would allow it to be easily incorporated into an overall cognitive system. This also implies that the structure of the resulting cognitive search algorithm would resemble that of the overall cognitive system.

Function optimization is easier to address because its parameters are clearly defined, can be based on real

numbers, a clear definition of a "solution" exists, and a large body of work exists on how to find a solution. The idea is to determine whether developing the search algorithm using a subset of cognitive structures, will cause the search to have and display characteristics, behaviors, and performance associated with human cognition; human cognition is not the gold standard, but it is far beyond machine cognition. Since these behaviors and performance are desired in both problem domains, it is assumed that the general techniques used to achieve them in one, will yield similar results when applied to the other. It is not expected that the details of the cognitive search algorithm can be directly applied in assistive living, but general guidelines can be discerned. These statements are difficult to prove, so arguments are presented along with specific examples to support them.

The rest of this paper gives some background on cognition and discusses how it can apply to assistive living environments. The aspects of cognition of interest are then presented, followed by an examination of the mapping between the two problem domains. Cognitive optimization is examined and it is shown how the use of the cognitive aspects of interest produces characteristics, behaviors, and the performance resembling human cognition. Guidelines taken from the development of the cognitive search algorithm are then presented along with a discussion of their application to the assistive living problem. The details involved in actually implementing an assistive living system will not be addressed.

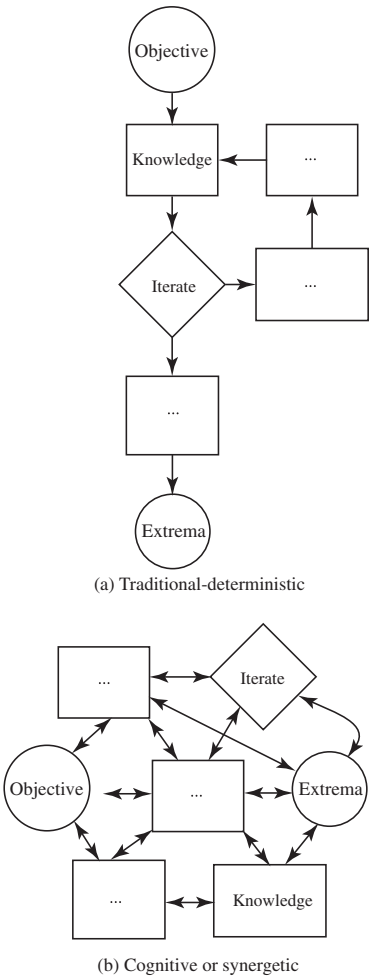

Fig. 1. Deterministic vs cognitive system.

2. Overview

2.1. What is cognition?

The aspects of cognition of interest are: 1) the use of inherited knowledge to arrive at a solution or action, 2) the use of communication between different system components to facilitate organization, and 3) synergism or self-organization, (hereafter, the adjective "cognitive" refers only to these aspects unless otherwise stated.) In contrast to synergism, a deterministic system relies on explicit definitions of logic paths between inputs and outputs, as shown in Fig. 1a. The cognitive approach uses a finite set of elements or processes, considered as inherited knowledge, which communicate in order to self-organize around a particular solution or action, as shown in Fig. 1b.

Communication involves message passing in order to share information with other processes. Messages can follow different paths through the processes, in effect, reorganizing them. Logic paths are generated from permutations of the processes, rather than from an explicit encoding of every possible path. In the psychology literature one can find several diagrams similar to Fig. 1b which map sections of the brain to different functions, and illustrate how these areas communicate with each other in order to self-organize around particular actions [13, 14]. This map can be simplified into four broad aspects of cognition, *perception*, *reason*, *decision*, and *action* operating in the assistive living apartment shown in the center of Fig. 2.

The cognitive approach allows a decomposition of the problem into simple processes whose inner

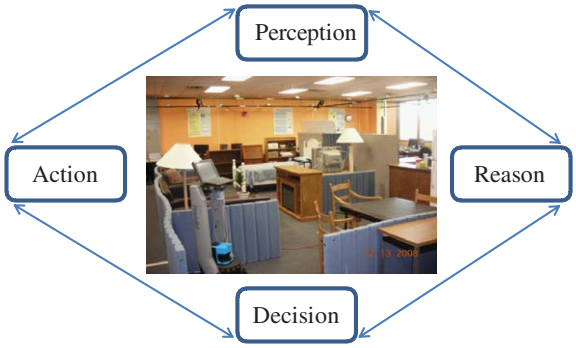

Fig. 2. Cognitive assistive living environment.

workings should be easy to understand, thereby simplifying the development of a cognitive system. This approach also allows different means for including knowledge in the system, rather than as a set rules that explicitly map inputs to actions. This work examines the cognitive optimization problem as a means for exploring how a subset of human cognitive abilities can be achieved in an assistive living system.

2.2. *Assistive living and cognitive optimization*

The assistive living system includes a monitoring system with a capability for affecting the environment, through robotics and automation. Perception is accomplished by a suite of sensors positioned throughout the apartment. The monitor's reasoning abilities must interpret the sensor data to determine what is happening in the dwelling. For example, the system may perform an analysis to determine if the occupant has fallen. If so, it must make some decision as to how to respond. This may include calling an attendant or deploying a robot. Once a decision has been made, the corresponding action must be executed. All elements in Fig. 2 can communicate with each other to self-organize into the appropriate action. Thus Fig. 2 should include more paths of communication than are shown.

It is a difficult task to directly develop a cognitive system for an assistive living environment. Thus, the idea is to develop a cognitive system for the simpler problem domain of function optimization, as shown in Fig. 3. This is driven by the notion that optimization is a key function in a cognitive system. In addition, two key performance criteria in function optimization are similar to those in an assistive living system, the *convergence rate* and *feasibility*.

In an assistive living environment the monitoring system should quickly determine a response to input data, especially in emergencies such as a fall. Human cognition can quickly discern whether a patient has fallen and determine a course of action. Similarly, in function optimization, search algorithms are evaluated by their rate of convergence to a solution. Feasibility is concerned with the remarkable ability of human cognition to find a feasible solution to a wide range of dissimilar problems. The assistive living system must also be able to handle numerous different problems and situations that occur in an assistive living environment, an ability that might be improved using a cognitive

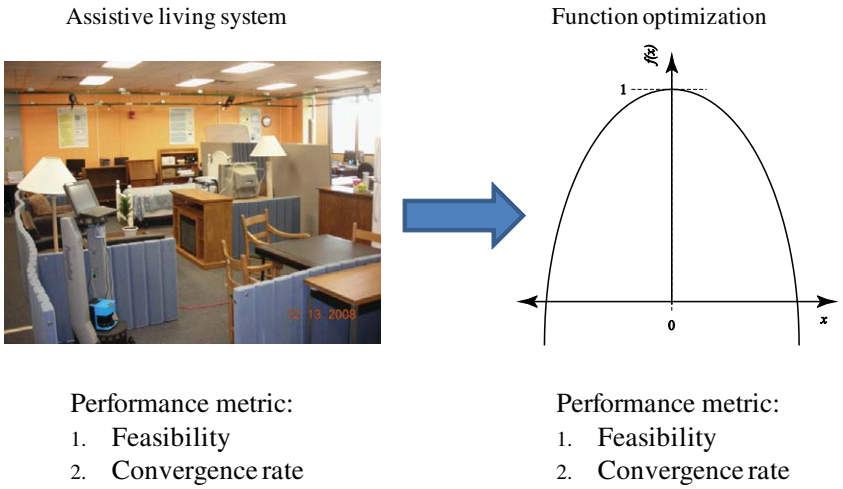

Fig. 3. Mapping between problem domains.

structure as opposed to a rule-based one, for example. Similarly, the cognitive search algorithm has a feasible problem set that is larger than the intersection of the feasible problem sets for the individual inherited search methods.

2.3. This paper's scope

The effort here is to develop a cognitive search algorithm which displays characteristics of human cognition in terms of its structure, convergence, and feasible problem set. However, there is no attempt to develop the most efficient, complete, comprehensive, or competitive cognitive search algorithm possible. The cognitive search is developed only to the extent that it reveals characteristics and behavior similar to human cognition. It is then analyzed to determine guidelines for the development of a cognitive assistive living system. Further developments in these areas will be pursued in future work.

3. Background on cognitive optimization

The cognitive approach can be categorized as a nontraditional optimization technique. Traditional optimization methods center around mathematical operations related to geometry, calculus, probability, statistics, and other mathematical formalisms. These approaches do well in problems where the function to be optimized, and the domain of the search, are continuous. The drawback is that they are only guaranteed to find local optima.

Nontraditional approaches are used in an effort to overcome these shortcomings. They are able to work on discontinuous functions over discontinuous domains and can find global optima. This is an increase in performance over traditional methods because it becomes feasible to address a new set of problems. Well-known examples of nontraditional optimization are referred to as *evolutionary approaches* such as the genetic algorithm. The idea behind the evolutionary approach is to randomly populate the function domain with a number of individuals that represent guesses at the solution. These individuals are perturbed at each step, creating subsequent generations that converge to the optimal solution, Kennedy [8].

The older of the evolutionary approaches, the genetic algorithms and its variations, mimic natural selection in nature. It begins with a randomly selected population of individuals whose fittest are bred to produce new generations such that the population's average fitness increases with each generation, Goldberg [5]. A mutation step is also involved where select individuals are randomly altered. Observations of social insect colonies led to a combination of natural selection and learning from social interaction which produced the *ant colony* [1, 3, 4] and *particle swarm* [2, 7, 9, 12, 19] optimization techniques. These techniques differ from the genetic algorithm in that each individual alters the solution it represents using information from others, with or without a breeding process.

Another step forward in nontraditional optimization is referred to as *social cognitive optimization* [11, 16–18] which is based more on human cognition. This algorithm implements a more sophisticated form of social interaction. The population of individuals are referred to as *social cognitive agents* which can learn from others and have their own memory. Rules of operation are encoded in each agent which can write its knowledge to a library that all agents have access to. It is the interplay between learning and memory that causes the algorithm to find an optimal solution. It also converges faster and with fewer function evaluations than the genetic algorithm or particle swarm [16, 17].

The work closest to the proposed approach involves the social cognitive agents in [16, 17]. The difference is their reliance on random sampling to learn about the function using techniques from machine learning; other non-traditional search algorithms, such as simulated annealing, also rely on random sampling. Each social cognitive agent has its own memory and can communicate with others, which makes them somewhat complex. *This complexity goes counter to the findings from the investigation presented herein.*

One of the goals here is to determine whether self-organization can be achieved without relying on the randomness of evolutionary, and most other nontraditional approaches. This is desirable because randomness can obscure the underlying characteristics of the search algorithm. Here the goal is to find this underlying structure, so randomness is omitted to facilitate its discovery.

4. The cognitive approach to optimization

The following developments determine whether utilizing a cognitive structure leads to cognitive behavior. A simple function,

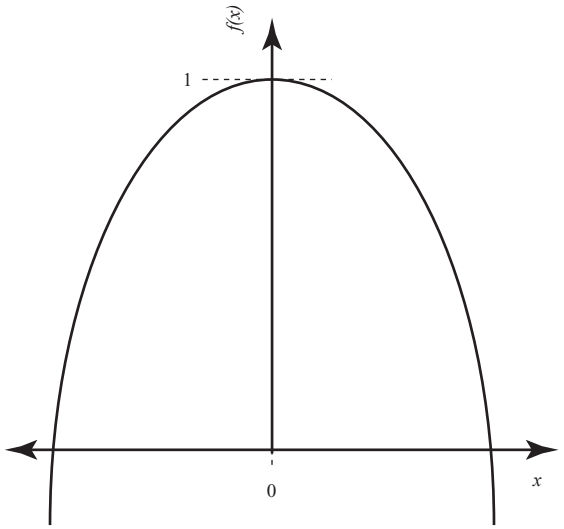

Fig. 4. $f(x) = 1 - x^2$. Properties: continuous, nonlinear, unconstrained, convex, single global optimum, univariable.

$$\max_x f(x) = 1 - x^2 \qquad \frac{df(x)}{dx} = -2x \qquad (1)$$

shown in Fig. 4, is used to initiate this investigation in order to avoid the complications and uncertainties raised by choosing a complex problem. The optimal solution is $(x, f(x)) = (0, 1)$. The first derivative equals zero at this solution, which will be used as the optimality condition for the cognitive search. The following sections address the three aspects of cognition and discuss what must be done to achieve the human-like performance sought.

4.1. Inherited knowledge

The inherited knowledge consists of two well-known search techniques, the *bracketing* and *gradient* algorithms. Two optimization algorithms are chosen that search in different ways. It has been suggested that using processes that function differently is a key element in a cognitive system that generates something like "imagination;" a random component can also fulfill this purpose, Werbos [15]. The difference between them is that the bracketing search jumps discontinuously around the function's domain, while the gradient search travels more continuously along its peaks and valleys.

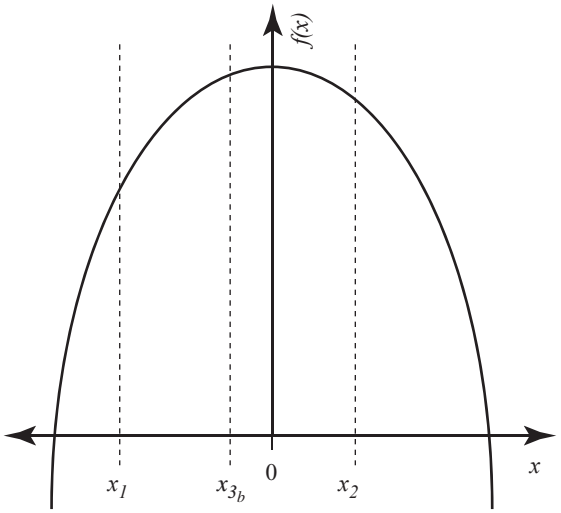

Fig. 5. Bracketing, $x_{3_b} = (x_1 + x_2)/2$.

4.1.1. Bracketing search

The bracketing search starts with two initial guesses that define a bracket or region that includes an extremum. This is true if the first derivatives of the initial guesses have opposite signs. The goal is to continually reduce the region within the bracket until it surrounds the extremum to within a given tolerance. This is accomplished by evaluating the midpoint and replacing the upper or lower bracket depending on the sign of its derivative. This bracketing search is different than the one traditionally used in root finding.

This process is shown in Fig. 5 where x_1 and x_2 are the initial guesses and x_{3_b} is the midpoint of the bracket. The first derivative at x_{3_b} is determined and if its sign equals that of x_1, then x_{3_b} replaces it in the bracket. The new bracket is defined between x_{3_b} and x_2, which more tightly surrounds the extremum. The next step would be to consider $x_{4_b} = (x_2 + x_{3_b})/2$ and so on until the region within the bracket becomes small enough to conclude that the extremum has been found.

4.1.2. Gradient search

This search uses the gradient, or first derivative, to find a candidate x which is closer to the extremum. The gradient gives the direction in which the function is most increasing, as shown in Fig. 6. If the initial guess is x_2, the next value to check is:

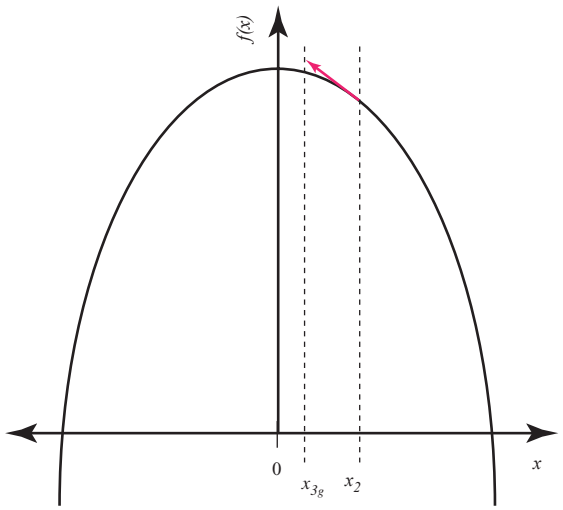

Fig. 6. Gradient Based Search.

$$x_{3_g} = x_2 + s\,\nabla_x f(x_2) \tag{2}$$

where $\nabla_x f(x_2)$ is the gradient of $f(x)$ evaluated at x_2, and s is an arbitrary number which moves x_{3_g} away from the initial guess x_2 in the direction of the gradient. *The performance of the gradient approach is highly dependent on the choice of s.* The gradient at x_{3_g} is used to find x_{4_g} and so on.

These techniques are used to formulate the marginally cognitive system shown in Fig. 7. The searches run independently and a new `Compare` process returns the solution with the largest function value. Each search retains its original structure, which can be as complex as a social cognitive agent.

Performance Comparison of the independent solutions must wait until both algorithms have finished. *Thus this cognitive system converges at the rate of the slowest search algorithm.* One might think that a simple way to improve performance would be to stop the cognitive search when the first search ends, but this is not effective since the other search might find a solution nearer the optimum. The system in Fig. 7 does not take advantage of the cognitive structure's power to achieve fast convergence.

4.2. Simple communication

Rather than independently running the two searches, communication between them can be increased to improve performance. In Fig. 7 it is assumed that each search independently checks the optimality of its

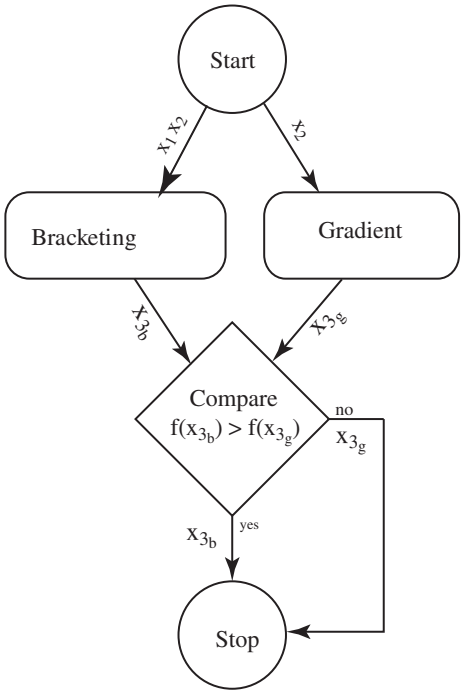

Fig. 7. Marginally cognitive optimization. Each process runs independently.

candidate solution. However, it is reasonable, and possibly more efficient, to remove this check from the individual searches and have an external function do it for both. Since optimality is usually checked after each iteration, it is reasonable to reduce the bracketing and gradient searches to single iteration processes; every optimization algorithm utilizes the notion of an iteration. Removing the optimality check simplifies both searches.

The revised procedure is shown in Fig. 8. The result of each iteration must be fed back into the algorithm so that it may continue the search; the search algorithms form an internal loop within the procedural flow of Fig. 8. The variables x_4 and x_5 define the new bracket after each iteration. In psychology these are referred to as re-entrant processes, because they accept their own output as input; re-entrant processes are an important characteristic of cognition, Thelen [13].

The result of each iteration is compared and the largest value is checked for optimality. Since a simple, continuous function is used here, its derivative can be used to check optimality. If the solution is optimal, the cognitive search terminates.

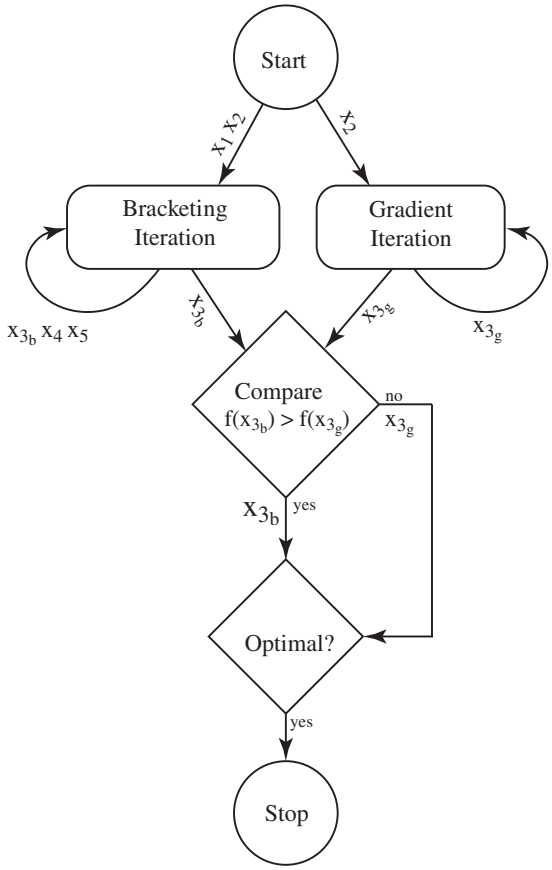

Fig. 8. Simple cognitive optimizer. Searches report solution after each iteration, but operate independently.

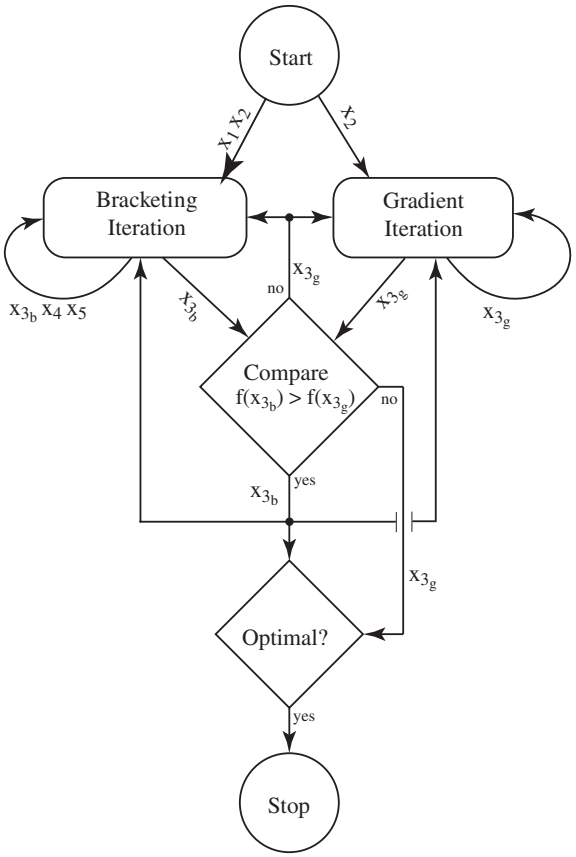

Fig. 9. Extensive communication cognitive optimizer. Communication occurs between the searches.

Performance *The procedure in Fig. 8 will converge at the rate of the fastest search algorithm.* This is an improvement over what was accomplished in Sec. 4.1., which converged at the rate of the slowest search algorithm. The searches still essentially run independently.

4.3. More extensive communication

In Secs 4.1 and 4.2 the processes did not really communicate with each other. To remedy this, the result from `Compare` is returned to both functions, as shown in Fig. 9; note that the procedural flow is beginning to resemble Fig. 1b. *The key is to determine how each search algorithm can best utilize this communication.* This is simple for the gradient search which can begin its next iteration at the value provided by `Compare`. The bracketing algorithm is a bit more difficult to adjust. Since the concept behind the bracketing search is to reduce the bracket's size, the incoming information is also used to reduce the bracket. This can be accomplished in several ways that can increase or decrease performance.

The most effective approach attempted essentially moves the bracket to surround the solution sent from `Compare`, and then performs its usual bracket reduction around the new point. The additional information allows the bracket to be reduced twice, about a point nearer to the optimal, in a single iteration. When `Compare` returns the value originating from the gradient search, the bracketing search attempts to collapse both sides of the bracket around the candidate solution; collapsing only one side of the bracket does not actually improve performance. Herein, if the bracketing search originated the value returned from `Compare`, the additional bracket reduction is not performed, thereby isolating the effect of communication.

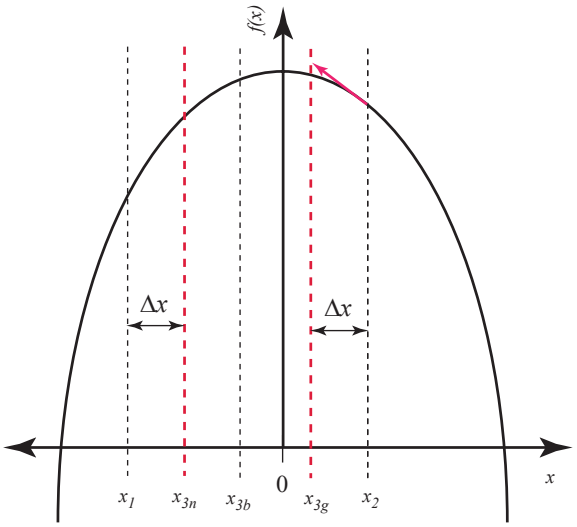

Fig. 10. New bracketing scheme.

This procedure is shown in Fig. 10 using a combination of the data in Figs. 5 and 6. The solution found by the gradient search, x_{3g}, is closer to the maximum, so `Compare` will pass it to the bracketing search, which immediately replaces the upper bracket, x_2, with x_{3g}. It then calculates the amount of this move, Δx, and attempts to pull the lower bracket, x_1 in by the same amount to x_{3n}, which reduces the bracketed region.

There are several ways to do this, but in practice good results were obtained with,

$$x_{3n} = \frac{(x_1 + \Delta x) + x_{3g}}{2} \tag{3}$$

which is checked to see if the slope of the function has changed between x_1 and x_{3n}; if it has, then point x_{3n} is discarded and the lower bracket is maintained at x_1. The process in (3) is not depicted in Fig. 10. The next candidate solution is $x_{4_g} = (x_{3g} + x_{3n})/2$.

Performance See Sec. 4.4

4.4. Synergism or self-organization

The processes shown in Fig. 9 have been simplified to the point where none can solve the problem alone. It is the communication between them that allows a solution; note that the re-entrant messages are considered to be communication. Thus these processes are significantly different than the social cognitive agents discussed in [16,17] where a single agent can solve the problem alone. The cognitive architecture allows the bracketing and gradient searches to cooperate in finding the extremum. It also allows the system to choose, or self-organize around, the process that works best in each situation as the search progresses. In addition, the original bracketing search performs well on this problem, so including it as inherited knowledge transfers its performance to the cognitive search.

Table 1
Bracket, gradient, and cognitive search performance

Method	Steps	Function evals		Time (sec)
		$f(x)$	$\frac{df(x)}{dx}$	
Gradient	194	196	390	1.15×10^{-2}
Bracketing	62	64	64	4.71×10^{-3}
Cognitive	49	128	213	6.74×10^{-3}

The processes in Fig. 9 should be run synchronously, but here they are run sequentially to facilitate a performance comparison between the bracketing and gradient search algorithms operating alone. In Table 1, one iteration of the cognitive optimization involves a combination of one iteration of the bracketing and gradient searches, one call to `Compare` and one to `Optimal?`. All examples in this paper were run on a DELL Precision T3400 with a 266 GHz Dual Core processor and 3.25 GB of RAM with $s = 0.1$ and a optimality tolerance of 2.22×10^{-14}. The optimizations in Table 1 were started with initial values of $x_1 = -632$ and $x_2 = 700$.

Performance *The system in Fig. 9 converges to a solution in fewer iterations than the bracketing or*

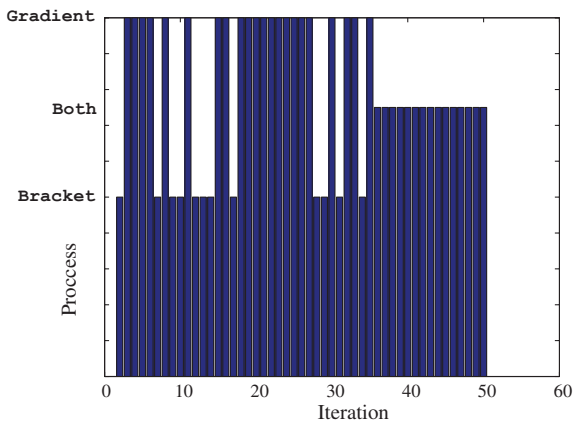

Fig. 11. Process originating the value returned by `Compare` at each iteration.

gradient searches alone, as shown in Table 1. When other performance measures are considered, such as the number of function evaluations and the run time, the cognitive approach lies in the middle. This is understandable since each iteration of the cognitive search involves evaluation of both the bracketing and search algorithms. Yet the cognitive approach is well suited for parallel processing, so it is possible to significantly reduce the overall run time if the communication does not introduce large delays. For this reason, the number of iterations is used as the performance metric.

Figure 11 shows the originator of the value returned from `Compare` at each iteration of the cognitive search. It continually switches between bracketing and gradient searches, except at the label `Both` which indicates when the difference between x_{3_b} and x_{3_g} is small enough that $f(x_{3_b})$ and $f(x_{3_g})$ are numerically indistinguishable; however, the first derivatives, evaluated in `Optimal?`, are distinguishable. No discernible pattern to this switching exists, implying that the cognitive search automatically adjusts to converge to a solution. *This implies that the cognitive search self-organizes around the optimal solution procedure.*

It is desirable to run more cases to determine whether this behavior is general, or occurs only for the case in Table 1 and Fig. 11. Thus 1000 trials were run where the initial guesses were randomly sampled from a uniformly distributed set such that $-1000 \leq x_1 \leq -10$ and $10 \leq x_2 \leq 1000$. The results of these trials are given in Table 2, which agree closely with Table 1. Notice that the mean for the bracketing algorithm is closer to its maximum number of iterations while its standard deviation is small. This means that most of the runs required a number of iterations clustered closely around 61. In contrast, the mean for the cognitive approach is closer to its minimum number of iterations 45 and the standard deviation is large. Thus most likely, most runs required a small number of iterations with some outliers requiring a large number of iterations.

Table 2
Statistical comparison of bracket, gradient, and cognitive search algorithms

	Gradient	Bracketing	Cognitive
Iterations			
max	196	63	66
avg ± dev	192 ± 16	61 ± 3	54 ± 11
min	176	50	45
run time (sec)			
max	0.011	0.0045	0.017
avg ± dev	$0.011 \pm 6.8 \times 10^{-8}$	$0.0034 \pm 1.2 \times 10^{-8}$	$0.0083 \times 7.1 \times 10^{-7}$
min	0.0096	0.0030	0.0065

4.5. Summary

The implementation of communication in Sec. 4.4 approaches the desired cognitive behavior in terms of faster convergence with respect to the number of iterations using inherited knowledge, communication, and self-organization. In addition, since no random sampling of the function was used, although a random process could be added, it is only the structure of the cognitive optimization that produces the highly desirable behavior. This indicates that the structure of the cognitive optimization may be approaching that of the underlying general cognition. Using this structure in a cognitive assistive living system could yield a similar fast convergence.

5. Increasing the feasible problem set

Sec 4. focused on the rapid convergence aspect of cognitive system performance. It is also desirable to examine whether the structure of the cognitive search can increase the number of problems that can be solved by the cognitive system beyond the intersection of what the bracketing and gradient searches can address alone. Having a large feasible problem set is similar to human cognition's ability to address a broad set of dissimilar problems.

Several evolutionary optimization techniques, such as the genetic algorithm, simulated annealing, etc., can address the problems discussed in this section. Incorporating these other techniques into the cognitive search would increase the feasible problem set of the cognitive search. This is not done because an original goal was to avoid using random elements in the search. Thus the focus here is to determine whether the cognitive structure can increase feasibility for the bracketing and gradient searches. Next, some optimization problems are considered that pose difficulties for the bracketing and gradient searches.

5.1. Multiple local maxima

Alone, the bracketing and gradient searches only find local optima; they cannot guarantee a solution for the global optimum. A function with multiple local maxima is shown in Fig. 12; the extrema

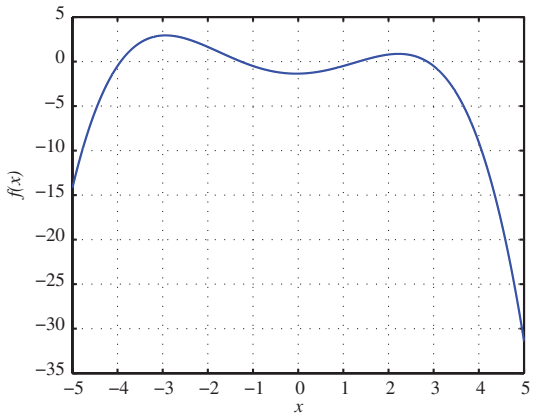

Fig. 12. $f(x) = -1/14(x+4)(x+1)(x-1)(x-3) - 0.5$. Properties: continuous, nonlinear, unconstrained, nonconvex, local optima, univariable.

are $(x,\ f(x)) \in \{(2.22, 0.86), (-2.94, 2.94)\}$. One can probably imagine several simple approaches that would allow determination of the global optimum. The key is to implement it without much change to the system in Fig. 9, since it already converges quickly for certain problems.

A simple solution stems from the *branch and bound* technique used in integer and mixed-integer programming [6]. It involves partitioning the domain into several brackets. For illustrative purposes, it is simplest to define and search two brackets using the procedure in Fig. 9. The user supplies two sets of upper and lower bounds, essentially more knowledge, which define the brackets. Note that the actual branch and bound technique automatically defines brackets on the fly, depending on the properties of the function. It is desired to eventually implement this functionality, but not in this paper.

Addressing several brackets simultaneously is equivalent to increasing the number of messages circulating in the cognitive search. This is shown in Fig. 13 where the initial brackets are x_1, x_2, x_3, and x_4 and the brackets after a single iteration are x_7, x_8, x_9, and x_{10}. The candidate optima after a single iteration are $x_{3_b}, x_{6_b}, x_{3_g}$, and x_{6_g}. The values returned by `Compare` are $x_{3_{max}} = max(x_{3_b}, x_{3_g})$, $x_{6_{max}} = max(x_{6_b}, x_{6_g})$, and $x_{max} = max(x_{3_{max}}, x_{6_{max}})$. This is similar to creating a population in a genetic algorithm, except with a smaller population whose size is simple to determine.

One iteration/step of the new cognitive algorithm consists of the original algorithm operating on each

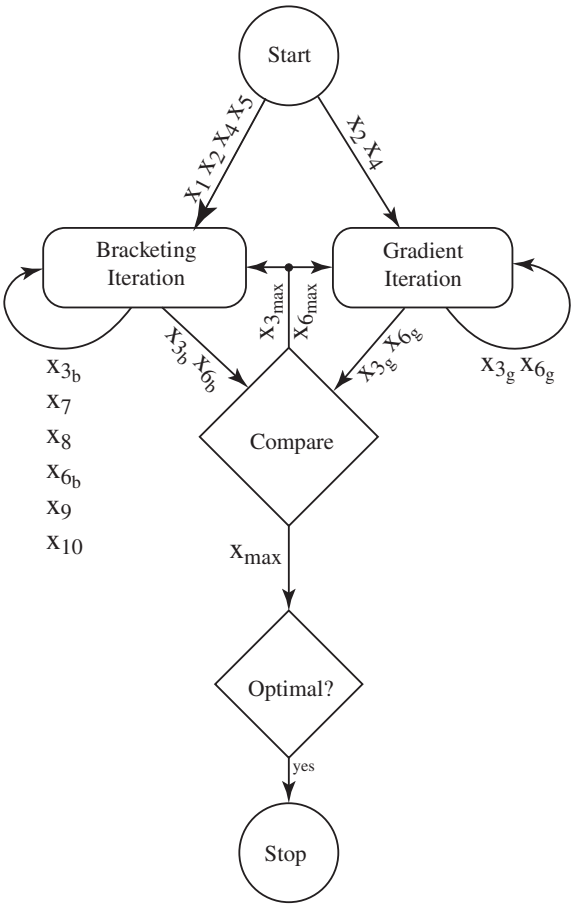

Fig. 13. Multiple bracket cognitive optimization. Increasing the number of messages can address multiple local optima.

bracket once. If a region does not contain an extremum, the bracket will close completely at one of its boundaries, otherwise it will find a local maximum. When all brackets have reached one or the other condition, the largest solution is returned. Still, the implementation in Fig. 13 can only guarantee that the global optimum is *more likely* to be found.

Performance Table 3 shows that the bracketing, gradient, and cognitive search from Fig. 9 can fail to find the global maximum depending on the initial guess, while the procedure in Fig. 13 has a better chance of finding it. The run time for the new cognitive algorithm is longer, but this is expected and necessary to locate the global optimum.

Fig. 14 shows which process and bracket originated the largest function value at each iteration. There

Table 3
Comparison of Bracket, Gradient, and Original and New Cognitive Search Algorithms.

Method	Initial guesses	Final answer	Steps	Evaluations		Time (sec)
				$f(x)$	$\frac{df(x)}{dx}$	
Gradient	(0, 50)	(2.22, 0.86)	91	93	184	5.24×10^{-3}
Bracketing	(0, 50)	(2.22, 0.86)	51	53	53	3.15×10^{-3}
Original cognitive	(0, 50)	(2.22, 0.86)	41	85	166	5.62×10^{-3}
New cognitive	(−100, 0, 50, 100)	(−2.94, 2.94)	53	322	611	4.78×10^{-2}

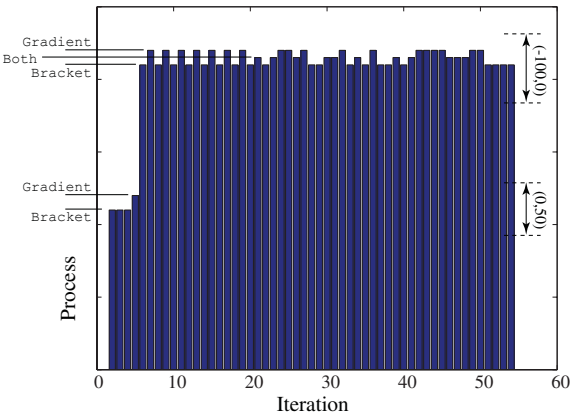

Fig. 14. Process and Bracket Originating x_{max} at each Iteration.

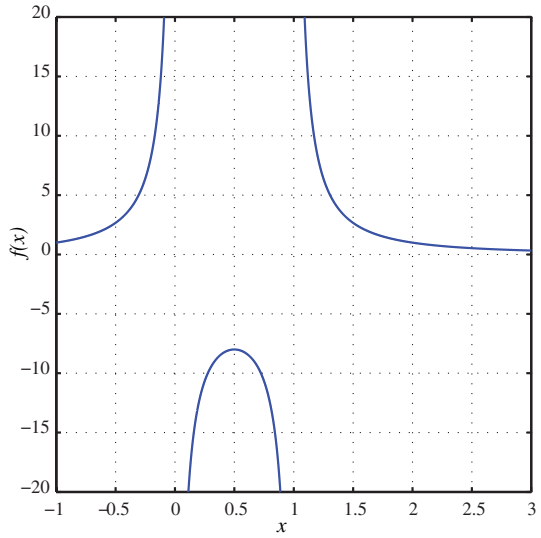

Fig. 15. $f(x) = 2/(x^2 - x)$. Properties: discontinuous, nonlinear, unconstrained, nonconvex, local optima, univariable.

is no discernible pattern of switching between the gradient and bracketing searches, so the property of self-organization has been retained. These results show that the feasible problem set has been increased beyond the intersection of these sets from the bracketing and gradient algorithms, in terms of the likelihood of finding the global optimum.

5.2. *Discontinuous functions*

A difficult maximization problem involves discontinuous functions such as the one shown in Fig. 15. The bracketing, gradient, and earlier cognitive searches can all fail to find the optimum depending on the initial condition, similar to what occurred in Sec. 5.1 Thus here no comparison is made with these earlier approaches. The intervals between the initial guesses $x \in \{-10, -1, 3, 10\}$ yield three brackets which are used with the approach depicted in Fig. 13; all initial guesses lie on an asymptote.

Performance In 510 iterations, using 1212 and 1905 function and derivative evaluations, the global optimum, $(x, f(x)) = (0.5, -8)$ was found in 0.12 seconds; more processing is required to solve more difficult problems, as expected. `Compare` returned the value originated by the gradient search 504 out of 505 iterations, as it repeatedly attempted to travel up the asymptotes; the bracketing search value was chosen at the final iteration. The re-entrant values, x_{3_b} and x_{6_b}, allowed the bracketing search to progress, essentially finding the optimum alone. This reinforces the notion that re-entrant communication is a key part of cognition [13]. *The cognitive search increases the likelihood of finding the global optimum, where the earlier approaches may fail.*

5.3. *Nonlinear constraints*

The problem in Sec. 5.2 is made more difficult by adding a nonlinear constraint $x^2 + f^2(x) = 15^2$ in Fig. 16. This problem can be addressed using Lagrange

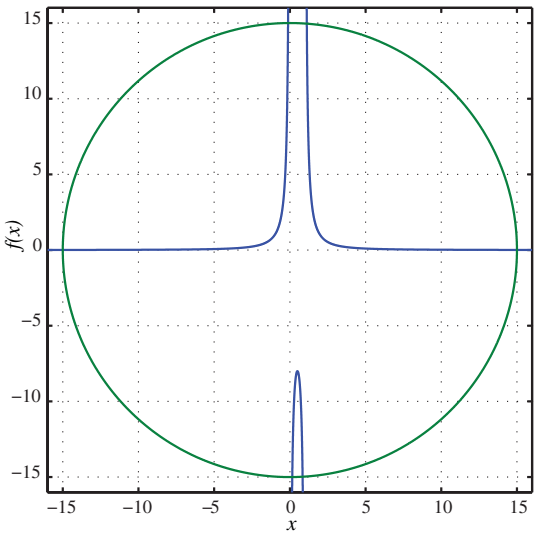

Fig. 16. $f(x) = 2/(x^2 - x)$ and $x^2 + f^2(x) = 15^2$. Properties: discontinuous, nonlinear, constrained, nonconvex, local optima, multivariable.

multipliers which allow a constrained problem to be converted into an unconstrained one with a larger number of variables:

$$\min_{x,\lambda} \bar{f}(x) := \frac{-2}{x^2 - x} + \lambda \left(x^2 + \frac{4}{(x^2 - x)^2} - 15^2 \right) \tag{4}$$

A candidate solution is found when the gradient of the function with respect to x and λ goes to zero. This multivariable problem has solutions along the asymptotes of $f(x)$.

The standalone bracketing and gradient searches were not altered to work alone with multiple variables, thus no comparisons will be made. Implementing the multivariable cognitive search required extensive additions to the procedure in Fig. 13. The details of these modifications are too lengthy to include here. Essentially, the process for a single variable was duplicated for the second variable. More processes and messages were added, including a process that would expand a bracket if it lost containment of an extremum. All of the original processes were retained, but they were modified to handle multiple variables and gradients. Four initial brackets were defined for each variable, the intervals between $x \in \{-100, -30, 10, 100\}$ and $-10 \leq \lambda \leq 10$, and independently adjusted based on the gradient of the function.

Performance In 66 iterations, and 301 and 713 function and derivative evaluations, the algorithm found the global optimum, $(x, \lambda, f(x)) = (-0.12, -0.033, 15)$ in 0.097 seconds. The value originating from the bracketing search was returned from `Compare` 64 iterations including the final result. This work was done to illustrate feasibility, but the algorithm requires more work to make it robust. *Therefore, a number of multivariable optimization problems are included in the cognitive searches feasible set to some extent.*

5.4. Summary

It has been shown that the performance of the cognitive search, in terms of convergence rate and feasibility, can be improved so that it appears possible to approach the performance of human cognition to some extent. In addition, no randomness was used to develop the cognitive search, so its resulting structure is the sole reason for the performance improvements obtained and the cognitive behavior observed. It appears promising that the general structure of the cognitive search algorithm, and the procedures involved in developing it, can yield a similar performance when used to develop a cognitive assistive living system.

6. General guidelines for cognitive system development

General guidelines that can be discerned from the development of the cognitive search algorithm in Secs. 4 and 5 are shown in Table 4. Other information can be discerned from the developments in Secs. 4 and 5, but this list will suffice until this process is explored

Table 4
Guidelines for cognitive system development

	Guideline
1	Processes should implement different methods to search for a solution
2	The component processes should be as simple and deterministic as possible
3	Processes should be formulated to fully utilize shared information
4	Processes should broadcast information as often as possible
5	Multiple messages should be passed and operated on simultaneously
6	Communication should be re-entrant
7	One or more processes must evaluate the system's progress

further. It is interesting that guideline 2 goes contrary to the notion of a social cognitive agent with complex capabilities.

7. Application to assistive living systems

The first thing to note is that the neat separation of the cognitive assistive living system into the four areas of perception, reason, decision, and action shown in Fig. 2 may not be realistic. This is because guidelines 2 and 3 imply that there will be numerous system components that are highly interdependent on information from each other. However, this makes sense because processes which only execute one basic function can be more easily rearranged to allow different responses to changing situations.

Guidelines 4, 5, and 6 imply that there must be some sophisticated and fast method for sharing information between processes. The amount of communication that may be required implies a decentralized structure to the processes. However, guideline 7 still implies some supervisory element which monitors the entire system to determine whether it is progressing toward a solution.

These guidelines can be further grounded by considering the structure of the cognitive system that would respond to a fall. One could write a procedure specifically to detect a fall that would include all of the parameters, logic, and data required. However, the guidelines in Sec. 6 suggest otherwise.

Consider the cognitive system in terms of the categories used in Fig. 2. The perception function consists of collecting sensor data. The guidelines suggest that several processes might be established to collect data from each sensor. They would also collect data from the same sensor using different approaches, sampling at different frequencies for example, or collecting different measurements from the same sensor. The data obtained should be broadcast, or made available, to all other processes in some manner. In addition, the processes associated with the sensors should accept requests to provide data on demand should it be needed by some other process.

The main reasoning involved in fall detection is determining the velocity of one or several reference points on the occupant's body. A number of simple processes should determine the velocity, in conjunction with other variables, using different approaches, to determine whether a fall has occurred. If a fall has occurred, the reasoning processes should trigger the rest of the system to determine some action.

This involves the decision processes choosing a course of action. Again these should be simple processes, that may need to request data from other processes in order to determine what to do; an occupant falling on the couch may require a different response than a fall in the shower. Simple processes can combine in different ways to respond to the subtleties between different falls and organize around the appropriate response without the having to write a separate rule to trigger the appropriate action for every possible fall scenario.

Once a decision is made the action processes need to carry it out. These should be simple processes that may check the current conditions to determine whether their associated action can be executed. Some may request additional data, or check if the hardware required for the action is functioning. Non-functioning hardware may require an alternate decision to be made. Some processes may monitor the occupant's current velocity to determine whether she/he stood back up, thereby ending the emergency and requirement for the action.

Some process must oversee the system in order to ensure something is done about the fall. It is not clear how this would function, but it may simply be a clock watcher that is triggered when the reasoning processes determine a fall has occurred. In fact, the details of this entire implementation are unclear, but this discussion should reveal how the different processes may communicate with each other in order to self-organize around a response appropriate for the severity and particular circumstances surrounding a fall.

8. Conclusions

This paper showed how some aspects of cognition can be implemented in a search algorithm in order to produce characteristics, behavior, and the performance associated with human cognition, and to yield guidelines for developing the same capabilities in an assistive living system. An interesting aspect of this work stems from the initial examination of how to best utilize the inherited knowledge in the search algorithm. Considerations of how to achieve performance with this knowledge began a cascade of reasoning that led to modifications aligning with the characteristics of cognition reported by psychologists. It was somewhat remarkable how this occurred, but it supports

the notion that the resulting structure of the search algorithm is typical for a cognitive system. This conclusion is strengthened by the fact that no random elements were used, which means the structure is solely responsible for the performance. Thus it likely that the guidelines found would apply to the development of any cognitive system including one intended to monitor an assistive living environment. Future work involves further development of a more robust and widely applicable cognitive search algorithm which may reveal additional insights into cognitive functionality. Further examination of the implementation details for use of these guidelines in an assistive living system will also be performed.

Acknowledgments

Many thanks to Rong Zhang of the Computer Science and Engineering Department at The University of Texas at Arlington for her advice on the implementation of the processes in the cognitive search algorithm.

References

[1] C. Blum, Ant colony optimization: Introduction and hybridizations. In: *Proceedings of the International Conference on Hybrid Intelligent Systems (HIS)*, September 2007, pp. 24–29.

[2] D. Bratton and J. Kennedy, Defining a standard for particle swarm optimization. In: *Proceedings of the IEEE Swarm Intelligence Symposium (SIS)*. April 2007, pp. 120–127.

[3] M. Dorigo and C. Blum, Ant colony optimization theory: A survey. *Theoretical Computer Science* 344(2–3) (2005), 243–278.

[4] W. Gao, Study on immunized ant colony optimization. In: *Proceedings of the International Conference on Natural Computation (ICNC)*. August 2007, pp. 792–796.

[5] DE. Goldberg, Genetic algorithms in search, optimization & machine learning. Addison-Wesley Publishing Company, Inc., 1989.

[6] FS. Hillier and GJ. Lieberman, Introduction to operations research. 4 ed. Holden-Day, Inc.

[7] X. Hu, Y. Shi and R. Eberhart, Recent Advances in Particle Swarm. In: *Proceedings of the Congress on Evolutionary Computation (CEC)*. June (2004), pp. 90–97.

[8] J. Kennedy, Review of Engelbrecht's Fundamentals of Computational Swarm Intelligence. Genetic Programming and Evolvable Machines **8**(1) (2007), 107–109.

[9] J. Kennedy and C. ER, Particle swarm optimization. In: *Proceedings of the International Conference on Nueral Networks*. November (1995), pp. 1942–1948.

[10] M. Looks, B. Goertzel and C. Pennachin, Novamente: An integrative architecture for general intelligence in *AAAI Fall Symposium - Technical Report*. October (2004), pp. 54–61.

[11] T. Ray and KM. Liew, Society and Civilization: An optimization algorithm based on the simulation of social behavior. *IEEE Transactions on Evolutionary Computation* 7(4) (2003), 386–396.

[12] MP. Song and GC. Gu, Research on Particle Swarm Optimization: A Review. In: *Proceedings of the International Conference on Machine Learning and Cybernetics*. August (2004), pp. 2236–2241.

[13] E. Thelen, in *The Dynamical Systems Approach to Cognition*. World Scientific Publishing Co. (Vol. 10) (2003) pp. 17–44.

[14] W. Tschacher and JP. Dauwalder, (eds). The dynamical systems approach to cognition. World Scientific (Studies of Nonlinear Phenomena in Life Sciences; vol. 10) (2003).

[15] PJ. Werbos, Intelligence in the brain: A theory of how it works and how to build it. *Neural Networks* 22(3) (2009), 200–212.

[16] XF. Xie and WJ. Zhang, Solving engineering design problems by social cognitive optimization, In: *Genetic and Evolutionary Computation - GECCO 2004*, Lecture Notes in Computer Science, Springer Berlin/Heidelberg, 2004, pp. 261–262.

[17] XF. Xie, WJ. Zhang and ZL. Yang, Social cognitive optimization for nonlinear programming problems. In: *Proceedings of the International Conference on Machine Learning and Cybernetics* (2002), pp. 779–783.

[18] YX. Yu and HP. Zhang, A social cognition model applied to general combinatorial optimization problem. In: *Proceedings of the International Conference on Machine Learning and Cybernetics*, Beijing, China, November (2002), pp. 1208–1213.

[19] Z. Zhang, R. Zhange and L. Xin, An improved particle swarm optimization algorithm with sentient mode. In: *Proceedings of the IEEE International Conference on Mechatronics and Automation (ICMA)*. August. Harbin, China, 2007, pp. 2342–2346.