Available online at www.sciencedirect.com
SCIENCE DIRECT®
Electronic Notes in Theoretical Computer Science
ELSEVIER
Electronic Notes in Theoretical Computer Science 141 (2005) 163–179
www.elsevier.com/locate/entcs

Adaptation and Evolution in Dynamic Persistent Environments

David Keil and Dina Goldin[1]

Computer Science & Engineering Dept.
University of Connecticut
Storrs, CT USA

Abstract

Optimization (adaptation) of agents interacting with *dynamic persistent environments* (DPEs) poses a separate class of problems from those of static optimization. Such environments must be incorporated into models of interactive computation.
By the No Free Lunch Theorem (NFLT), no general-purpose function-optimization *algorithm* can exist that is superior to random search. But *interactive* adaptation in environments with persistent state falls outside the scope of the NFLT, and useful general-purpose interactive optimization *protocols* for DPEs exist, as we show.
Persistence of state supports *indirect interaction*. Based on the observation that *mutual causation* is inherent to interactive computation, and on the key role of persistent state in multiagent systems, we establish that indirect interaction is essential to multiagent systems (MASs).
This work will be useful to researchers in coordination, evolutionary computation, and design of multiagent and adaptive systems.

Keywords: adaptive systems, coordination, dynamic persistent environments, evolutionary computation, interactive computing, models of computation, multiagent systems

1 Introduction

Environments of adaptive or intelligent agents have been characterized along five dimensions: (1) accessible vs. inaccessible; (2) deterministic vs. nondeterministic; (3) episodic vs. nonepisodic; (4) static vs. dynamic; (5) discrete vs. continuous [28]. The most difficult environments for which to develop intelligent systems are those that approximate the real world, i.e. ones that are

[1] Email: {dkeil,dqg}@engr.uconn.edu

1571-0661/$ – see front matter
doi:10.1016/j.entcs.2005.05.021

inaccessible, non-deterministic, nonepisodic, dynamic, and continuous. For function-based computation, captured by Turing machines, the properties of an environment are of no consequence, because one execution of an algorithm simply computes a function on whatever single input arrives from the environment. (Nondeterministic Turing machines compute functions that yield *sets* of outputs.)

Some work on *evolutionary computation* has characterized real-world environments in a similar way and has pointed out that an agent's environment may be a function of the existing agent population [13]. By contrast, assumptions about the environment (e.g., *constraints* on it) are part of the interactive *problem specification*. To design cars, for example, we assume they will be driving on a paved road, with gravity and temperature conditions normal on Earth. We may further restrict the environment for research purposes by assuming that the road has lane dividers and is clear of obstacles.

Evolutionary computation emerged to address difficult optimization problems using interactive processes of selection, mutation, and recombination found in nature [15]. In accordance with the *function-based* paradigm that dominates computer science, however, EC has traditionally been conceived as the *algorithmic* search for solutions to *algorithmic* problems. Thus, the "evolutionary algorithm" is applied to static "function optimization" problems. In Section 2, we show how this traditional way of posing problems leads into paradoxical results such as the No Free Lunch Theorem that seem (falsely) to lead to pessimistic conclusions.

Models suitable for evolutionary computation in real-world-like environments must incorporate the environment [20,10]. A significant research trend redefines the computational problem from one of static optimization to one in dynamic environments [8,5]. We suggest a focus on *persistent state* (memory) in such environments. It is persistence of state that enables non-episodic behavior. Our interest is broader than evolutionary computation, because the lessons that can be learned about *dynamic persistent environments* (DPEs) generalize to any adaptive system.

By the No Free Lunch Theorem (NFLT) no function-optimization algorithm exists superior to random choice. For dynamic persistent environments, however, we show the existence of useful general-purpose optimization protocols (Section 3). For example, life forms and human societies have survived in dynamic environments by *learning* methods that have broad applicability.

Because of the power of persistence, the problem of adaptation to a persistent environment may become more challenging than for nonpersistent environments. Since persistent state supports indirect interaction among agents with access to that state, the question of multiagent systems (MASs) arises

naturally in DPEs.

This paper suggests some new research directions and novel conceptual frameworks, offering several contributions to a multidisciplinary theory of multiagent interaction and multiagent systems:

- We define *interaction* in a way that recognizes the *mutual causation* among agents that exists in all truly interactive systems (Definition 2.1).
- We show that the No Free Lunch Theorem (NFLT) is not applicable to optimization of interactive behavior in DPEs (Section 3).
- We show that multiagent interaction with persistent state entails the use of *indirect* interaction (Theorem 4.5).

We note a gap between the informal setting or motivation in this paper and the formal and technical parts, which are of narrower scope. The latter aspect is a research challenge that we only begin to attack here. We believe that providing proper motivation and scope for the problem at hand is a contribution in itself.

2 Interaction and dynamic persistent environments

Here we identify mutual causality as an essential characteristic of interaction and define dynamic environments with persistent state.

2.1 Interactive computation

Interaction is a form of computation in which communication occurs *during* the computing process rather than only *before* or *after* [18]. The semantics of interaction entail mutual causality.

Definition 2.1 Interactive computation *is the ongoing exchange of data among participants (agents or their environment) such that the output of each participant may causally influence its later inputs.* □

Exchange of data that never affects the actions of the recipient, and never has a later effect on the inputs of the sender, is not true interaction. A person who responds to what is shown on a television screen, by talking to or shouting at the television, is not interacting with it. A microphone and an amplifier do not interact unless *feedback* is present. Research in cybernetics fifty years ago recognized feedback or mutual influence as a feature that distinguishes an important kind of coupling of systems [2].

Note that the outputs of two agents may causally influence their later inputs without the agents communicating directly; they may communicate

via an intermediary. In this case the causality and interaction are *indirect* (Section 4).

An *algorithm* is a description of the steps for effectively transforming an input to an output, where the output is a (computable) function of the input [22]; the environment plays no role in this description since all information about it is presumed to be captured in the input. By contrast, in interactive computation, the role of the environment is heightened; it is an active partner in the computation.

Definition 2.2 *The* environment *of an agent is the set of entities that it interacts with.* □

Communication between a single agent and its environment defines a variety of interaction:

Definition 2.3 *A* sequential interactive computation *continuously interacts with its environment by alternately accepting an input string and computing a corresponding output string [17].* □

Sequential interactive computation is interaction involving only two participants. Each participant may be considered the *environment* of the other. Sequential interactive computation may be formalized by *Persistent Turing Machines* [17]. Many other models exist, such as the model for agents operating within an environment presented in [31].

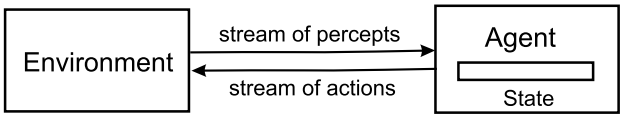

Fig. 1. Agent and environment

In Section 4, we will define indirect interaction formally and will show that it may be identified with multiagent interaction, an alternative to sequential interaction.

2.2 Dynamic environments with persistent state

For a computing agent's actions (outputs) to affect any later percepts (inputs) other than immediate ones, that agent's environment must have a *persistent* but alterable *state* on which the percepts depend. This notion is consistent with Piaget's definition of *behavior*, whose purpose is to change the state of the environment [26].

For function-based computation, captured by Turing machines, the properties of an environment (such as persistence) are of no consequence, because one execution of an algorithm simply computes a function on whatever sin-

gle input arrives from the environment. For interactive computation, on the other hand, how an action by a computing agent will change its environment is crucial to any choice made by the agent.

Let us define some *properties* of environments that are of interest to us. *Persistent environments* are those that *can* remember from previous interactions:

Definition 2.4 *An environment is* persistent *w.r.t. an agent if its outputs to the agent depend on the agent's earlier actions within it. An environment is* amnesic *if it is not persistent.* □

An electric light that lights when it is off and the user presses its button switch, but turns off when it is on and the user presses the same button, defines a persistent environment with respect to the user. A piece of paper defines a persistent environment w.r.t. a person writing on the paper, but is amnesic w.r.t. a person who is only reading it. The air is amnesic w.r.t. a person singing to it, and persistent w.r.t. a person spraying perfume into it.

Definition 2.5 *An environment E is* static *with respect to agent A if A's inputs from E are strictly dependent on A's outputs to E. A* dynamic *environment is one that is not static.* □

A lamp that lights dependably when the button switch is pressed and goes out when pressed again defines a static environment w.r.t. the person operating the lamp. A lamp with a light sensor, which lights when switched on only if the room is dark, defines a dynamic environment with respect to the user.

Any environment that can be modeled by a Turing machine is static, because whatever output an agent in such an environment emits, the environment will respond with a value that is a (computable) function of the agent's output. Such an environment remembers nothing from previous interactions, just as every computation by a Turing machine begins with a blank tape, not with results left on the tape from previous computations. An objective function, such as is used in evolutionary computation research, defines a static environment.

Our concern is with environments that combine dynamism with persistence. A *dynamic persistent environment (DPE)* with respect to an agent is one that is *persistent* but not *static* with respect to it. A DPE, E, interacts with agent, M, in such a way that M's actions may change the state of E, affecting M's later inputs.

For example, the environment in which a car drives is dynamic and persistent (see [11] for a discussion of the car driving problem). It changes with respect to the car whenever the car moves. Cars, pedestrians, and other poten-

tial obstacles appear and vanish in a way that is partially determined by the car's actions. One model for the type of environment we describe as dynamic and persistent is Markov decision processes (MDPs), particularly partially observable MDPs [19].

Recent expressiveness results for models of interactive computation enable us to show that dynamic persistent environments are *harder to adapt to* than dynamic environments without persistent state.

Definition 2.6 *A class A of environments is* more difficult *than a class B, with respect to set S of agents, iff A's range of observable behaviors is a strict superset of B's.* □

Lemma 2.7 *The set of system behaviors observable in DPEs strictly includes the set of behaviors observable in amnesic environments.* □

Proof sketch: We may formalize a DPE as a Persistent Turing machine with state [17]. Amnesic environments may be formalized as amnesic PTMs, those that can store or remember nothing or that make no use of stored information. By a theorem in [17], the set $\mathcal{PSL}$ of stream languages of PTMs strictly includes the set $\mathcal{ASL}$ of stream languages of amnesic PTMs.

It follows immediately from Definition 2.6 and Lemma 2.7 that DPEs are more difficult to adapt to than amnesic environments.

In the next section, we show that the notion of dynamic persistent environments can help evolutionary-computation research escape from some apparent theoretical impasses, related to the problem of adaptation.

3 Where the No Free Lunch theorem does not apply

The No Free Lunch theorem (NFLT) [30] presented a paradox to researchers in evolutionary computation (EC). On the one hand, by the NFLT, no algorithm (such as an evolutionary algorithm) performs more efficiently than random search at solving the general function-optimization problem. On the other hand, natural evolution has proved better at producing organisms that attain fitness in varied environments than random search could have done. We solve this paradox by showing that the NFLT does not apply to optimization in DPEs, such as natural environments.

3.1 A paradox for evolutionary-computation research

In simple terms, we have by the NFLT that no *algorithmic* procedure optimizes cost functions more efficiently than any other, when the algorithms' efficiencies are averaged over the set of *all* cost functions [30]. For example, if a certain evolutionary algorithm may converge quickly to solve a certain optimization

problem (the lunch) then for some other problem (the price of lunch) this algorithm performs much worse than random search. Thus as a universal problem solver, no EA is better than random search.

Formally, the NFLT establishes that for any two algorithms a_1 and a_2, and any cost function f, the following equality holds:

$$\sum_f P(\vec{c} \mid f, m, a_1) = \sum_f P(\vec{c} \mid f, m, a_2)$$

where $\vec{c}$ is the histogram of values an algorithm obtains for f given m evaluations of f. The probability expression $P(\vec{c} \mid f, m, a_i), i \in \{1, 2\}$ is the conditional probability of that histogram [30].

It follows from the NFLT that to build a good function optimizer one needs to know something about the function beforehand. The proof of the NFLT has withstood inspection, but nature seems to have has produced a counter example, in that *life on earth* appears to be performing general-purpose (if slow) optimization, via evolution [6], and *humans* appear to be doing so as well, at a faster pace, via intelligence. It is paradoxical that despite the restriction imposed by the NFLT, fit adaptive systems come into being all the time in generalized processes that make no a priori assumptions about the environment.

Some research has succeeded by stricting severely the class of cost function sets under consideration. The performance of some algorithms, averaged over a finite set F of cost functions, may be *better* than that of other algorithms if the set F is not closed under permutation (c.u.p.). Furthermore, almost all sets F are not c.u.p. The use of restrictions or prior knowledge of fitness landscapes in virtually all EC research lends further weight to the idea that even for the class of "realistic" cost functions, no good universal optimization method exists.

3.2 Resolving the paradox

The NFLT rules out algorithmic procedures for *general* optimization. It follows from the NFLT that every useful evolutionary algorithm is to some degree problem specific in that it has some useful built-in knowledge of the fitness landscape. Natural and artificial systems gain the necessary knowledge by *interacting* with their environments, sometimes over millennia.

A way out of the dilemma is indicated by the fact that the theorem stated above defines a_1 and a_2 as *algorithms*, not interactive *policies*, and the set of possible environments considered by the theorem are *functions*, not first-class objects that may evolve in interaction with populations of solutions.

Algorithms can't explore environments, adapt in response to inputs, or change their behavior; execution of an algorithm begins only after acquisition of all inputs.

The notion of dynamic persistent environments offers a solution to the paradox. In such environments, the cost function for an agent or population may change throughout an optimization process. Function f is replaced by a *sequence* of cost functions f_t. Since evolution in DPEs is not an *algorithm* optimizing a *function*, the NFLT does not apply to optimization in DPEs, whether natural or artificial. Both an agent and its environment are *reactive systems* in the sense of [23]. It has been noted that the design of agents that are reactive systems operating in reactive environments is concerned with utility maximization, rather than with satisfaction of predicates as in the design of functional systems [31].

To put it another way, as life forms evolve, they change both themselves and their environments. In contrast, evolutionary algorithms don't do this. For example, the NFLT does not apply in the following contexts:

- As *microorganisms* evolve in the ocean, changing the ocean's conditions, they tend to optimize their population's ability to survive; as they interact with their ocean environment, the cost function changes over time, partly as a result of the population's development.
- As a *market* evolves, it tends to optimize quantities produced and prices, but via *interaction* among agents in the economy, not through an *algorithm*.

Unlike human-directed optimization processes, natural evolution lacks a global purpose; moreover, species, and even often organisms, lack the will or mental intention to survive. Nevertheless, processes of natural evolution tend to optimize the capacity of individuals of a species to survive.

The significance attributed to the NFLT among EC researchers reflects the *mathematical world view* that sees problems as *functions*, i.e., as transformations of single inputs to single outputs [18]. To acknowledge the importance of DPEs is to depart from this traditional mathematical world view and to embrace a broader, interactive world view.

The well-known EC researcher Kenneth DeJong observed that effective strategies for function optimization are not necessarily effective for sequential decision problems, and reminded us that the motivation for John Holland's foundational work on EC was a concern for adaptive systems. Evolution in general is a way "to explore and adapt to complex and time-varying fitness landscapes" [7].

A conceptualization of computing that incorporates interaction would suggest that an agent seeking to minimize or maximize a function should *learn*

interactively about this evolving cost function and the mechanism inducing it. Whereas no algorithm can embody knowledge of all cost functions, an interactive process can be constructed to learn and to guide itself by interacting with environments that induce cost functions, including costs that change dynamically in interaction with the process. Some such interactive *learning* processes will perform better than blind search. This fact explains the existence of complex life forms.

The facts that natural evolution on Earth has obtained better results than random search would have done, and that many forms of life are themselves better general-purpose optimizers than random search, are proofs that processes exist that perform more efficiently at finding extrema than random search. Since they are not *algorithmic* processes, the NFLT does not rule out their existence.

The observation that *coadaptation* (mutual adaptation) between agents and their environments *molds* fitness landscapes in a favorable way [12] highlights the role of interaction in permitting escape from the unfavorable implications of the NFLT. In fact, it is precisely coevolution of species that refutes the claim by William Dembski that the NFLT implies a necessary role for intelligent design in accounting for complex life forms [25]. Coevolution and coadaptation by definition define, for their participants, environments that are dynamic and persistent.

The NFLT sets limits on our ability to find optimal general-purpose *algorithms*, as the Church-Turing Thesis helps define limits of algorithmic computation. Just as [17] suggests an interactive extension to the Church-Turing thesis, a challenge for research in a theory of interactive computation is to provide a formal negation of the NFLT, including counter-examples, in such a way as to account for the existence of useful general-purpose optimization protocols in interactive contexts.

3.3 Toward formal results

We present our conclusions about the NFLT and DPEs semi-formally, as two observations and a conjecture that all follow from the above discussion.

> *Observation*: Problems of optimization for interactive computation with dynamic persistent environments cannot in general be reduced to function-optimization problems.

Let an environment E interact with computing agent A in such a way that A's inputs from E are not a function of A's immediately preceding outputs; they may also be affected by E's state, as shaped by A's previous outputs.

Thus the problem of optimizing A's inputs over time is not one of optimization of a function from input to output. Another parameter is the changing state of the environment.

> *Observation*: Consider the set of all environments that have existed on earth since the cooling of the earth's crust. The evolution and survival of life forms lends support for the idea that selection, recombination, and mutation of DNA, as the basis for the genesis and reproduction of organisms, constitute a robust protocol for the generation of survivable life forms in such a set of environments.

The natural history of the earth's crust over several billion years confirms this observation. Let us then venture a guess about an artificial protocol with similar general effectiveness.

> *Conjecture*: Let π be the protocol that consists of developing and refining a model of the environment and using that evolving model to predict and obtain desired inputs interactively from the environment (it could be described as the scientific method plus good engineering practices). Then π approaches optimality for all environments encountered by humans so far or in the foreseeable future. π is better than random choice for obtaining results suitable for human survival and comfort.

Clearly a theorem analogous to NFL, but describing the limits of optimization in DPEs, could be developed, because the easiest DPEs for which to optimize cost functions are amnesic (memoryless) ones, equivalent to static environments, i.e., for which interactive optimization is equivalent to function optimization. However, it is also clear that only environments whose behavior can somehow be characterized as *reasonable* are of interest; a tiny fraction of all conceivable environments. In particular, it may well be that the only dynamic environments that are of any practical interest are ones that are *learnable*, i.e., environments for which a *computable* model can be developed that aids in optimizing reward in interaction with the environment.

4 Indirect interaction and multiagent systems

Persistence of state in an environment offers agents within it a way to interact with each other via that environment instead of directly. We define *indirect interaction* and show that it is distinct from, and not reducible to, direct interaction (Section 4.1). Section 4.2 offers examples of this in nature, illustrating the multidisciplinary aspect of the research area of multiagent systems. We show that it is precisely indirect interaction that distinguishes multiagent

interaction from sequential interaction (Section 4.3). If systems featuring indirect interaction may display a greater range of behavior than ones without, then DPEs with indirect interaction (multiagent DPEs) pose more challenging problems than those with only direct interaction. Hence multiagent solutions are required.

4.1 Direct versus indirect interaction

It is commonly thought that interaction is the same as communication. Communication is associated with the notion of *message passing* [24] or *targeted send/receive* (TSR) [14]; we refer to it as *direct interaction.*

Definition 4.1 Direct interaction *is interaction via messages; the identifier of the recipient is specified in the message.* □

Message passing is appropriate when two entities possess identifying information about each other. Lacking such information, however, communicating entities may communicate anonymously by altering the persistent state of their common environment. This is a second category of interaction.

Definition 4.2 Indirect interaction *is interaction via persistent, observable state changes in a common environment; recipients are any agents that will observe these changes [20].* □

Example 4.3 *(The Dining Philosophers problem) [9]*
In this classic problem in concurrency and shared resources, philosophers sit in a circle to eat, with one chopstick between each pair of diners. Diners individually pick up two chopsticks (one at a time), eat, put them down, and think, repeating the steps endlessly. The problem is to avoid starvation by deadlock; if each diner uniformly picks up the left (or right) chopstick, for example, and holds it until the other one is available, then all will starve.

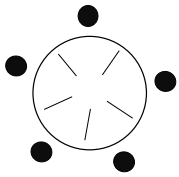

Fig. 2. Dining Philosophers

The objective of this problem is to define a protocol for a ring-shaped arrangement of communicating processes, each communicating only with its two neighbors, such that all processes are allowed to move forward under the constraint that no two adjacent ones may execute simultaneously. Here, the chopsticks serve as parts of the environment (shared data) whose persistent state

(on-table or with-diner) enables them to be used as a medium of communication, resulting in indirect interaction between neighboring philosophers. □

Since indirect interaction relies on persistence of the state of the environment over time, the identity of the recipient of communicated information is determined by dynamic (late) binding. The decoupling between sender and receiver in indirect interaction implies *anonymity* and *asynchrony*. Anonymous interaction occurs when computing entities communicate without knowledge of each other's identities. Asynchrony in indirect interaction follows from time delay due to the use of persistence of the observable changes over time.

Alongside anonymity and asynchrony, indirect interaction typically has the features of *locality* and *non-intentionality* [16]. The former means that efficient indirect interaction allows the agent access (for either perceiving or modifying) to only a small part of the environment. The latter means that participants may interact without any pre-existing goal or intention of communication. That is, as the agent makes changes to the environment that are to be perceived by others, he may be acting for reasons that include no explicit intention to have his changes thus perceived.

Example 4.4 *(The Dining Philosophers problem, continued)*
This problem has the properties of (1) locality*, because diners can only see neighboring chopsticks; (2)* anonymity*, because diners need not know each other's names or even whether their neighbors exist; (3)* asynchrony*, because diners don't necessarily pick up a chopstick as soon as it is put down; and (4)* non-intentionality *of communication, because diners pick up chopsticks to eat with and put them down to think, not to communicate.*

Dining Philosophers is an example of a DPE problem. The environment *of each philosopher is the table and other diners around it; the* state *of the environment is persistent, changing in a way not controllable by any one diner.*

Dining Philosophers has also been modeled with direct interaction between philosophers and utensils, as in concurrency theory. In this model, chopsticks are autonomous computing processes, like philosophers. However, this contradicts the semantics *of this problem, whereby chopsticks are not autonomous entities. They are passive, initiating no action, and their only roles are to reflect the states of the philosophers next to them. The semantics of the problem are of interaction among diners, not between diners and utensils. Such interaction is indirect, via chopsticks; a model based on direct interaction would no longer exhibit the four characteristic properties discussed above.*

□

Because of the properties discussed above, indirect interaction is natural for agents whose cognitive resources, such as those to plan, to perceive and

to act upon their environment, are limited relative to the complexity of their task [32].

4.2 Indirect interaction in nature (stigmergy)

Almost all real-world optimization problems are problems of adaptation by multiagent (multi-component) systems, or MASs, to dynamic persistent environments. Adaptation in MASs may be *centralized*, based on rational deduction that yields algorithmic solutions. Or, in the absence of rational problem-solving and planning mechanisms, it may be *decentralized*. Examples of decentralized adaptation include ants, termites, and slime mold.

In the StarLogo *termite* simulation, the termites build a circular pile of wood chips, despite having no capacity for planning or coordination, and with minimal ability to perceive. They continuously apply a simple protocol: move at random, pick up a chip whenever one is encountered, and put it down when the termite bumps into another chip. Eventually, a single pile emerges. This global behavior of the termite population is more than the composition of the individual chip-carrying behaviors. The termites accomplish this pile-forming task, in a self-sustaining manner, without an internal representation of the goal [27].

Ant colonies solve the problem of efficiently foraging for food sources by a decentralized multiagent interaction in which each ant deposits pheromones (evaporating scent chemicals) as it walks, and each ant follows pheromone trails as well as food odors. Heavily traveled (hence strong, hence attractive) pheromone trails correspond to short paths to food. As food is exhausted at a site, the trails to it evaporate. Without a plan, the ants find a set of paths to the food that tends toward optimality [3,4].

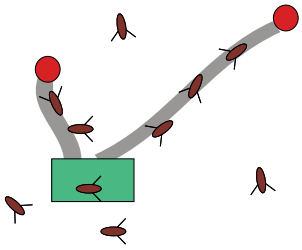

Fig. 3. Ant foraging

When their food is scarce, *slime mold* amoeba organisms gravitate to one another by use of a chemical signal emitted into the environment. The signal is relayed among the organisms, which migrate toward the center of a spiral of such signals, eventually aggregating into a single crawling slug-like organism [21]. Again, this aggregate behavior emerges from individual interaction

without centralized direction.

All these cases in nature are *multiagent systems*, as opposed to ones featuring sequential interaction (Definition 2.3). A common feature of the above examples is that organisms interact via their *shared environment* rather than by exchanging messages directly.

4.3 Multiagent systems require indirect interaction

Multiagent interaction (MAI) is ubiquitous, whether at the atomic, molecular, cell, organism, social, or planetary level. Although computer science models computation as transformation of input to output by one computing agent, and although concurrency theory models interaction as direct communication between two agents or processes, complex systems in the real world are richer than either model. Systems are often *open* in the sense that entities or streams of interaction may be created or destroyed. *Asynchrony* and *anonymity* of communication have brought into being the field of *coordination models* for MASs [1,14].

Models of MAI that represent only the direct and synchronous interaction of pairs of agents show significant limitations. Because MAI is more than the composition of multiple instances of sequential interaction, a model of multiagent interaction is required that reflects its special character. MAI is different from sequential interaction precisely in that it involves indirect interaction. In the following, we seek to put the distinction between sequential and multiagent interaction in sharp relief by proving the identification of MAI with indirect interaction. Intuitively, this identification is obvious. Yet models of concurrency to date do not explicitly represent this form of interaction; [24] models all interaction under concurrency as *direct* targeted send and receive (TSR).

Theorem 4.5 *Let S be a multiagent system that contains some agents and possibly shared variables that form a DPE w.r.t. the other agents. Then indirect interaction occurs in S.* □

Proof.

1. Let S be a multiagent system in which some subset of agents, and possibly shared variables, form a dynamic persistent environment with respect to the others.

2. Suppose S is characterized by direct interaction only.

3. By Definition 2.1 (interaction), a pair of agents in S interacts only if each affects its own later inputs by its actions. But by Definition 4.2 (indirect interaction), no pair of agents in S communicates via state changes created by one and observable by the other in a common environment.

4. Hence no part of S has persistent state, hence no part can be a DPE w.r.t. the other part (Definition 2.4), contradicting (1). So (2) has led to a contradiction and must be rejected.

[29] surveys the field of environments for MASs, making frequent reference to indirect interaction. A special category of MAS is those that are *decentralized* but cooperating in a common task. For a discussion of MDPs controlled by multiple cooperating agents, including via indirect interaction, see [19]. It presents a model of decentralized partially observable Markov decision processes (POMDPs) that highlights the use of indirect interaction.

Indirect interaction is necessary for full expressiveness of MAS models, but not sufficient. Indirect interaction can occur without greater expressiveness; e.g., if the component agents are not truly concurrent. Note that any modular algorithm can be transformed to a MAS by replacing subroutines with agents and input/output relations with shared variables (indirect interaction). But clearly despite the indirect interaction, there is no increased range of behavior in that case.

5 Conclusion

In this paper, we have identified interaction with mutual causation and defined the class of environments that are dynamic and persistent.

Introducing the notion of dynamic persistent environments suggests new research directions for multiple disciplines and new ways of thinking about agents and interactive computation. It may suggest ways forward for research in evolutionary computation and adaptive systems, for example. As we have shown, recognizing that adaptation to dynamic persistent environments is not a kind of function optimization allows us to resolve the dilemma of the No Free Lunch Theorem.

Using the observation that mutual causality is a defining property of interaction, we have shown that multiagent systems with persistence necessarily feature indirect interaction. It follows that new models are required that incorporate indirect interaction explicitly.

Future research challenges include:

- showing that adaptive agents can perform better in DPEs if they have persistent state;
- formalizing observations about the NFLT and DPEs (Section 3.3);
- proving the greater expressiveness of models of multiagent/indirect interaction over models of sequential interaction; and
- formalizing multiagent systems in a way that explicitly incorporates indi-

rect interaction, a key notion for solving problems in dynamic persistent environments.

Acknowledgement: we thank the anonymous referees for valuable suggestions and references.

References

[1] Farhad Arbab. Reo: A channel-based coordination model for component composition. *CWI Report SEN-0203*, 2002.

[2] W. Ross Ashby. *An introduction to cybernetics*. University Paperbacks, 1964.

[3] Eric Bonabeau, Marco Dorigo, and Guy Theraulaz. *Swarm intelligence: From natural to artificial systems*. Oxford Univ. Press, 1999.

[4] Eric Bonabeau and Guy Theraulaz. Swarm smarts. *Scientific American*, pages 72–74, March 2000.

[5] Juergen Branke. *Evolutionary optimization in dynamic environments*. Kluwer, 2002.

[6] Joseph C. Culberson. On the futility of blind search: An algorithmic view of 'no free lunch'. *Evolutionary Computation*, 6(2):109–127, 1998.

[7] Kenneth DeJong. Genetic algorithms are NOT function optimizers. *Foundations of Genetic Algorithms 2, D. Whitley, ed., Morgan Kaufmann*, 1993.

[8] Kenneth DeJong. Evolutionary computation: a unified overview. *Tutorial at CEC-2001, Seoul, Korea*, 2001.

[9] Edsger Dijkstra. Hierarchical ordering of sequential processes. *Acta Inform.*, 1:115–138, 1971.

[10] Eugene Eberbach. On expressiveness of evolutionary computation: Is EC algorithmic? *Proc. World Conf. on Computational Intelligence*, 2002.

[11] Eugene Eberbach, Dina Goldin, and Peter Wegner. Turing's Ideas and Models of Computation. *In Christof Teuscher, ed.,* Alan Turing: Life and Legacy of a Great Thinker*, Springer*, 2004.

[12] Gary William Flake. *The Computational Beauty of Nature*. MIT Press, 1999.

[13] David B. Fogel, Lawrence J. Fogel, and Wirt Atmar. Hierarchic methods of evolutionary programming. *Proc. 1st Ann. Conf. on Evolutionary Programming*, pages 175–182, 1992.

[14] D. Gelernter and N. Carriero. Coordination languages and their significance. *CACM*, 35(2):97–107, 1992.

[15] David E. Goldberg. *The design of innovation: Lessons from and for competent genetic algorithms*. Kluwer, 2002.

[16] Dina Goldin and David Keil. Toward Domain-Independent Formalization of Indirect Interaction. *TAPOCS Workshop, Proc. WET ICE 04*, 2004.

[17] Dina Goldin, Scott A. Smolka, Paul Attie, and Elaine Sonderegger. Turing machines, transition systems, and interaction. *Information and Computation Journal*, 194(2):101–128, Nov. 2004.

[18] Dina Goldin and Peter Wegner. The Church-Turing Thesis: Breaking the Myth. *LNCS 3526, Springer*, pages 152–168, June 2005.

[19] C.V. Goldman and S. Zilberstein. Decentralized Control of Cooperative Systems: Categorization and Complexity Analysis. *J. Artificial Intelligence Research*, 22:143–174, 2004.

[20] David Keil and Dina Goldin. Modeling indirect interaction in open computational systems. *TAPOCS Workshop, Proc. WET ICE 03*, 2003.

[21] James Kennedy and Russell Eberhart. *Swarm intelligence*. Morgan Kaufmann, 2001.

[22] Donald E. Knuth. *The art of computer programming, Vol. 1: Fundamental algorithms*. Addison-Wesley, 1968.

[23] Zohar Manna and Amir Pnueli. *The Temporal Logic of Reactive and Concurrent Systems: Specification*. Springer Verlag, 1992.

[24] Robin Milner. Elements of Interaction. *Comm. ACM*, 36(1):78–89, 1993.

[25] H. Allen Orr. Devolution: Why intelligent design isn't. *New Yorker*, pages 40–52, May 30 2005.

[26] Jean Piaget. *Behavior and evolution*. Pantheon, 1978.

[27] Mitchel Resnick. Turtles, Termites, and Traffic Jams. *MIT Press*, 1994.

[28] Stuart Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. Addison-Wesley, 1995.

[29] D. Weyns, H. Parunak, F. Michel, T. Holvoet, and J. Ferber. Environments for multiagent systems: State-of-the-art and research challenges. *LNCS 3374*, pages 1–48, 2005.

[30] David H. Wolpert and William G. Macready. No free lunch theorems for search. *Santa Fe Institute technical report SFI-TR—0*, 10, 1996.

[31] Michael Wooldridge. On the Sources of Complexity in Agent Design. *Applied Artificial Intelligence*, 14(7):623–644, 2000.

[32] Franco Zambonelli and H. Van Dyke Parunak. Signs of a revolution in computer science and software engineering. *Proc. ESAW02*, pages 13–28, 2002.