Deep Blue is still an infant.

Robert Levinson
Jeff Wilkinson
Department of Computer Science, University of California, Santa Cruz, CA 95060
E-mail:levinson@cse.ucsc.edu
Phone: 408-459-2087

Abstract

We believe that by the time of the workshop Deep Blue will have lost another match to Garry Kasparov, showing little improvement over the previous one. But even if it is indeed a "new kind of intelligence", it can be argued that this intelligence is very basic. There is a long way to go until Deep Blue could be considered a fully autonomous agent, the type we are eagerly trying to build in AI and that we believe we will eventually be successful at.

We should not consider Deep Blue (or any software agent) fully autonomous until it can manage its own computational resources (memory and time), minimize risk and cost of various decisions, assess its errors and develop new representations of (chess) knowledge, and be able to cooperate and communicate with other computer and human chess players.

We feel indeed that Deep Blue falls somewhere non-trivial on the scale of intelligence. But to move further along that scale greater autonomy will be required: Is Deep Blue now selecting its own openings, deciding when and whether to play for a draw, etc? Does it generalize from its past experiences, annotate its own games etc.? Does it know that is has played the agent "GKasparov" before? That the match will last 6 games (like last time) etc. Can it apply the information or decision theoretic model on which it is choosing its moves to other operations management and control situations? Does Deep Blue know that one mistake against the World Champion could be fatal? Does it know that bishops can only reach one color square? Does it contain a declarative representation of the rules of chess?

The Next Generation of Chess Computers

While we readily acknowledge that the brute force approaches have clearly and significantly outshined the more knowledge-oriented approaches, we feel that down the road, advances will need to be made to integrate chess computers with modern AI, and the coming hi-tech information technologies. Although computer chess has advanced significantly in recent years many issues have barely been touched on:

- Current computer chess systems typically do not manage their own computational resources (time and space) well. Resources are either allocated and coordinated by hand, or the problem is largely ignored.
- The economic aspects of computational resources and information in terms of utility, risk, and cost are not appreciated and hence have been underexploited.
- Potential synergy among multiple statistical modules needs to be considered. The potential of a combination of both shared and adaptive chess analysis and learning has not been adequately examined.
- Knowledge recoding and compression is not considered since it is not required to play a game of chess itself - it is required, however, in order to operate in the world of chess information, analysis and literature. Will Deep Blue ever write "My System II"? Within any given chess program, even if some learning is employed, progress beyond a particular representation scheme is limited. Representational rigidity prevents automated generalization from previously represented positions and games to new but structurally related situations.
- The full expressive power of graphs, structural relations, and their use in analogical reasoning is not employed (due to presumed intractability). This cripples the mathematical power of an evolutionary chess program.

Resource thriftiness, shared data analysis and learning, representation change, and maintenance of varying expressivity are fundamental tasks required for creating fully autonomous and communicating chess agents. A data analysis and representation learning process will need to occur at progressively deeper levels of structural relationship.

To allocate resources (features or subevaluations) successfully, chess analysis methods should categorize or approximate the worth and reliability as they are encountered. Not only must they learn the characteristics

of individual resources, but also what it is reliably possible to accomplish when using them in concert. They should also examine and predict the combined efficacy of the resources at their disposal, and choose an analysis strategy based on this prediction and the acceptable level of risk. Externally provided or internally developed analysis tools should be selectively employed according to their demonstrated effectiveness in recognized contexts, and supplied with appropriate levels of computational resources. This is superior to using any single tool alone, using all tools, or using the right tool but using too much or too little of the available resources to meet the desired level of accuracy.

Meta-reasoning may be described literally as "reasoning about reasoning"(Russell & Wefald 1991). When meta-reasoning is applied to an underlying analytical procedure, the behavior of the procedure may be modified according to outcome of the meta-reasoning process. The objective when performing meta-reasoning is typically to improve the efficiency with which the underlying analytical procedure makes use of computational resources to achieve a set goal. Ideally, the additional resources invested in meta-reasoning will be outweighed by the resources that are saved in execution of the underlying procedure. If the same goal is achieved at a lower cost (or a better goal achieved at the same cost), the meta-reasoning may be called successful. Meta reasoning may even occur recursively, when reasoning occurs about whether or not to perform meta-reasoning.

We believe that is possible that a new generation of chess tools will be developed that operate synergistically with other tools to provide cost-efficient and reliable analysis for chess and have the capabilities to in addition support real-world decision making. As such systems develop, the research community should obtain an even better understanding of the effectiveness of Meta-Reasoning, coupled with analogical and structural representation, as an organizing principle for already powerful statistical methods.

In short, *if formulated properly, the same decision-theoretic, information-theoretic and economic considerations that lead to strong computer chess, can be applied to other domains as well, provided that these considerations are raised to a level in which they are made explicit (and modifiable) in a program's design rather than buried implicitly in a chess evaluation function and a search strategy. Given the increasing speed of computers this additional meta-level of interpretation should become feasible.*

Characteristics of an Ideal Agent

We find it useful to define the goal of AI research to be to build in software (mainly) the "best" complex adaptive system(Gell-Mann 1994) possible from a systems-theoretic perspective. Here we go over 10 characteristics of complex systems as outlined by Kauffmann and determine what it might mean for a chess computer while playing chess and a more general AI agent to have these characteristics. Each of these characteristics brings the system closer to the goal of *autonomy* , perhaps the key characteristic of intelligence.

Self-Stabilizing

Complex systems can keep themselves stable i.e. not break, or have performance diminish in a wide variety of circumstances. Deep Blue may have these characteristics with respect to chess positions (although there are probably some artificial endgame composition type positions in which it may not have a clue how to proceed) but as it is currently confined to playing chess and has limited sensory apparatus it would be at a loss in almost any other setting. Even playing a variant of chess without reprogramming, may be enough to induce weak performance.

Future AI systems will require detailed pattern recognition and sophisticated input systems to maintain stability in a wide range of environments.

Goal-seeking

The actions of complex systems appear purposeful. Certainly, in the chess arena Deep Blue exhibits purposeful behavior probably to a greater degree than any other chess computer, thus leading Garry Kasparov to call it a new kind of "intelligence". But can it play variants of chess with other objectives? Can it explain its activities in terms of subgoals? For example, "I am doing this to stabilize the king side, before I try to open lines on the queen side". More importantly, can it determine for itself its own goals and work towards them? Probably not. In fact, it is likely that Deep Blue did not select its own opening moves for the match or even whether to play for a win or a draw.

Program-Following

Complex systems can follow a sequence of instructions, even those involving branching based on conditionals. If Deep Blue's deep combinatorial search can not be construed as program following, we do not know what could have this property! On the other hand, Deep Blue currently does not have the ability to execute (without thinking) specific chess sequences such as the Bxh7 sacrifice, followed by Ng5, Qh5 etc. such compilation and execution of plans (higher order sequences of moves) are necessary for efficiency in calculations. For example, a human master can directly plan a path (and visualize it taken) for its king in the endgame and ignore countless other possibilities. Also, as Deep Blue probably uses only a few different evaluation functions and search strategies for an entire game of chess it may be missing opportunities to adapt to (i.e branch) specific themes on the board.

Self-Reprogramming

"A mouse may go down a lot of blind alleys while searching for cheese in a maze; after several trials, how-

ever, it will modify its search program and go directly for the cheese with few wrong turns" (Draper L. Kauffman 1980). Perhaps, the biggest disappointment of current chess computers is that they generally do not learn from their experiences. Some forms of rote learning now are being used (Scherzer, Scherzer, & Tjaden 1990), but on the whole, hundreds and thousands of hours of computation are being wasted by chess computers not caching and memoizing the results of their searches for future games. Every strong player knows a certain set of tactical motifs and combinations that recur frequently on the chess board. Unfortunately, chess computers are faced with discovering these motifs fresh when they sit down to each move.

During the match it is likely that whatever improvement or learning Deep Blue showed during the match was due to reprogramming or parameter changing instigated by the programmers rather than the machine itself. Since the team's understanding of chess is certainly weaker than the World Champion's we doubt such adjustments did much good, if at all. However, the machine, due to its computational power and accuracy, has the potential to achieve a level of chess understanding far exceeding that of humans, provided it is given a mechanism for compiling, summarizing and reusing lessons gained from experience.

Anticipation

"If you ring a bell one minute before each time you feed your dog, the dog will learn to associate the two. Soon, it will start to salivate—, that is, its mouth will literally start to water - whenever it hears the bell, even if there is no food around." (Draper L. Kauffman 1980). Deep Blue's "salivation ability" is currently only as good as the depth of its search, and the evaluation function and patterns that its programmers have presupplied it with. Without the ability to generalize and exploit patterns across positions, the machine's anticipation capabilities will never be stronger than they are right now (i.e. unless the programmers change the software or hardware).

This lack of accurate anticipation in certain circumstances is clearly Deep Blue's biggest weakness when compared to humans. Murray Campbell, one of DB's programmers, states: " Computers do not become tired or distracted. There is no psychology at work. If Deep Blue makes an error at all, it will only become clear later on in the game. This statement should be probably be modified to "it will only become clear to *it* later on the game" – a professional grandmaster can often recognize moves that create lasting weaknesses as they occur.

Environment Modifying

Deep Blue does not "modify its environment" – excepting of course for its position on a chess board. The day is probably still in the future where we will let machines have total responsibility for modifying their own physical environments. But steps can be taken in that direction. By using Metareasoning and learning from its past experience a machine should be able to allocate and request or trade for the resources it requires to do a given computation. How much control does Deep Blue currently have of the time it is using on its chess clock or on the size of its various caches or the depths of its search? Whatever assumptions have been made in implementing such changes should be made explicit. The economical and utility-based theory that a system is using to manage its own resources should be readily available. Currently too many AI agents have these most important details buried deeply in their code.

Perhaps Deep Blue should be given more responsibility for determining the utility of those people working on it and the training strategies adopted. Again these decisions are possible given the proper implementation of meta-reasoning.

Self-Replicating

Ideally, one would like AI agents and software of any kind to have the ability to create copies of themselves with improvements occurring in future generations, much as evolution makes use of biological systems. Ways in which software can be improved, while doing exactly the same thing (or better) include:

- Compilation in faster code.
- Compression into less code.
- Compilation into more explainable/modifiable code.

As much of Deep Blue is fixed, being built out of domain-specific hardware, it is difficult to imagine how such recompilations may be possible. But by making the decision-theoretic and information assumptions behind Deep Blue declarative they then become available for improvement through direct compilation into future versions. Indeed , Deep Blue should be able to simultaneously simulate various versions of itself and then evolve in the direction that appears most promising. This is probably done to a certain extent as an evaluation function is being tuned, but is not taking place on a large scale in which we can say a "new strategic entity has emerged...".

Self-Maintaining and Repairing

Certainly, the Deep Blue team only wishes that the machine was self-maintaining and repairing! However, the reality is that the machine does not contain a model of itself, and, sadly, probably is unaware (in the sense that it would make a difference) that it is playing "chess." There may be some redundancy built into Deep Blue, but it is unlikely that, on its own, it will be able to recover from a "bug" in its software - because it has no notion of what it is to be accomplishing.

In the past, computer chess programmers have excused losses as due to one bug or another introduced into the program. However, in the next generation as we move to building more robust and reliable complex

AI agents such "excuses" should be (ahem) inexcusable.

On a much more concrete level, Deep Blue almost certainly does not have a notion of redundancy applied to chess positions - as in the value of overprotection explained by Nimzowitch, and also probably does not have a notion of "repairing" any given aspect of a chess position.

Self-Reorganizing

As AI agents develop they will need to be able to make increasingly effective use of the resources available to them - including those they are already using. Part of an AI agent's responsibilities should include the envisioning of new uses and connections involving its existing mechanisms. "Most of us have had the experience , for example, of having bits of information that we've known for a while suddenly fit together to create a new picture of a situation. ... as we get older we are constantly rearranging our memories in more efficient patterns and even reorganizing our own personality system" (Draper L. Kauffman 1980).

The ability to recode, compress and reformulate its chess knowledge and experience would make Deep Blue a feared agent indeed. It is possible that through mathematical analysis by a machine, as versed in number theory and combinatorial games as the best humans, that new insights into the actual structure of the game itself could be discovered, exploited, and then explained by the machine. This is the direction of machine expertise we must pursue.

Self-Programming

Does Deep Blue know C? Imagine the power of a deep searching engine such as Deep Blue coupled with the constructs of a programming language such as C. With the ability to create and test myriads of subroutines and possibly put them to its own use in creating further ones there is no end to the creations of such an entity. Given the right principles and sound metareasoning framework in the original design I believe that such evolution is not only possible but will one day lead to a new label on computer processes, "DESIGNED BY COMPUTER".

A Simple Design of a Complex Adaptive AI System

In this section we outline the design of a complex adaptive system for chess-playing and other activities based on a decision-tree (hierarchy) of complex adaptive agents that employ meta-reasoning to manage their resources.

There are well known algorithms (Langley 1996) for converting from tabular representations (e.g. K-Nearest Neighbor tables (Omohundro 1987)) to hierarchical representations. Decision Trees are graph structures which branch from a distinct root node. From a starting node, such as the root, child nodes are connected. All nodes of the structure may have only one parent node, but may have many child nodes. No node (with the possible exception of the root itself) is the parent of the root node. This is what makes the root distinct.

A decision tree is employed by beginning at the root node and evaluating some logical functions. The outcomes of these functions determines the child node to which attention travels. As long as there are child nodes to the current node, more predicates are evaluated, and attention branches to the designated child node. Upon arriving at a "leaf" node from which there are no further branches, a decision is returned.

The relatively compact structure and rapid access time of decision trees make them an attractive format for re-representing learned utility data. Also, useful features and relationships may be easier to recognize in tree format, assisting the learning agent in generating more powerful representations.

There are variations on decision trees, such as regression trees that return continuous-valued results, rather than discrete decisions. Another variation, which we will employ here, is the interruptible or "any-time" decision tree, which maintains a default value at each node rather than just having values at the leaves. Such a tree may be traversed safely by an agent that has limited resources with which to perform the computation of predicates at each node. At any time, should enough resources not be available for the agent to perform the computations, the default value at the current mode will be returned. We regard the task of having each node optimize the choice of its default value under resource constraints, to be non-trivial and an excellent testbed for the design of advanced learning agents.

Structural Overview of a Functional System of Agents

A hypothetical design begins by considering a "base domain", an analytical problem domain meeting a small set of constraints, such as presenting finite lists of action-options over a sequence of discrete turns, and occasionally providing a continuous valued "performance measure". The domain may be simple or as complex as chess. Consider any relatively simple learning system, which employs raw, tabular forms of data storage as well as a straightforward algorithm for analysis. For our purposes, we will assume the use of KNN lookup tables. This system is designated the "base learner". A "boundedly rational base learner" follows the same design, but must complete calculations without consuming too much of restricted quantities of computational resources. Several extensions to this basic design are possible. We suggest examination of a hierarchically structured system of reinforcement learners. The speculative discussion below should provides the reader with a high-level overview of how we see such a system operating.

The boundedly rational base learner may also include any-time decision trees as additional knowledge sources. Both the trees and the tables are provided with limited computational resources for executing their evaluations. The knowledge sources that demonstrate themselves most able to provide statistically more predictive results and/or consume smaller amounts of computational resources will be provided with more resources for their development. This multi-source learner's performance in the base domain will be improved by good performance in the "meta-domain" of optimizing the resource allocation patterns of its knowledge sources.

Because of the fairly loose restrictions on our choice of base domain, it is possible to apply the same type of multi-source learning system to the domain defined by the resource usage optimization "micro-domain" Since the agents which maintain the tree node values themselves use tables and trees, a second meta-domain may be derived.

Based on the high level of generality of this system, we expect to observation analogical mappings between successful patterns and structures developed in the first meta-domain (derived from the management of knowledge sources in some base-domain) and those developed in the second meta-domain (derived from the management of knowledge sources in the micro-domain). The statistical detection (under resource constraints) of such analogical regularities describes a "macro-domain". Once again, the domain fits the loose requirements, allowing application of the same learning system, and the derivation of a third meta-domain. Again, we expect to observe structures and patterns that can be analogically related to those in the other meta-domains.

In all three meta-domains described, learners must cope with bounded rationality, which makes information a commodity. We have shown above that the ability to meta-reason about the relative supplies of computation resources to the various sources can lead to several abstract domains, which might at first appear to lead to intractability or inefficiency. However, the prospect that the successful strategies learned in different meta-domains might be mapped analogically from domain to domain offers hope for progress. Solving problems in one domain may solve problems in all domains. Making advances in any part of the learning system may be translatable into improvement at all levels of functioning.

Summary and Conclusions

It can be argued that problems in any of the meta-domains cannot be solved without solving the problems in all the domains at once. Here we outline how the proposed design addresses six critical (and open) issues in developing autonomous adaptive systems.

Resource Coordination: Self-management of computational resources is the central theme in each of the meta-domains. Success in any of the meta-domains will improve accuracy and/or resource consumption in the underlying domain

Economic Valuation: Underexploitation of decision theoretic considerations is avoided by using concepts such as the "gain ratio" (Russell & Norvig 1995) of a computation: the reduction in the entropy of the return value function divided by a measure of the resources expended. The generality of such information theoretic heuristics allow them to be applied at any level of a system.

Module Sharing: The development of the knowledge sources, in any of the domains, need not be performed with all sources in isolation. Interactions between the sources may be both competitive and cooperative within a single domain. The development of functional interaction schemes improves the efficiency of meta-reasoning in all domains.

Knowledge Compression: Representational change must be built in from the lowest levels to allow for efficient adaptation to novel domains. The detection of predictive features in domain states and the recoding of the domain representation to include these features improves the efficiency of all further computations. When information and computation become economic commodities, representational change is a capital investment upon which economic growth depends.

Representational Flexibility: Just as representational features must be readily transmutable, entire structures of representation must be open to recoding into alternative representations. Detection of commonly occuring domain structure features makes analogical mappings between the different domains possible. It is for this reason that we view representational changes as fundamental processes underlying advanced learning systems. Without it, information products cannot be imported/exported between domains.

Structural Reasoning: Logical relations can be represented as sets of boolean functions. Boolean functions can be represented as decision trees. Decision trees are graph structures. In turn, graph structures can be represented as logical relations specifying the connections between nodes. Transformations around this representational orbit can bring different structures into common forms, allowing the detection of regular structures.

Improvements in any of the meta-domains described above should produce improvements in the performance in each of the underlying domains. If the underlying domain is the original (practical) base domain, the improvement in performance will be direct. If the underlying domain is the micro-domain of setting default values, the improvement will be in performance of the any-time decision trees. If the underlying domain

is the macro-domain of detecting successful regularities in the two other meta domains, the improvement will be in the meta-reasoning process itself. All improvements should eventually lead to improved performance in the practical domain. *In summary, excellent chess play (the base domain) will, hopefully, arise out of a deeper understanding of the economics and decision-theoretic management of resources (including at various meta-levels, rooks or kings in chess, learning data structures, computational time and computer memory, and other agents or subagents) and this knowledge will be in a declarative form for future modification.*

Ongoing work

Our research group is exploring a variety of projects and themes that we hope point in the direction of the next chess computers and advanced AI. Our relevant papers on Computer Chess, meta-reasoning and games (Levinson & Wilkinson 1997; Allen, Hamilton, & Levinson 1996; Levinson 1996; 1992; Epstein & Levinson 1993; Levinson 1989; Levinson & Snyder 1991; Levinson 1993; Levinson *et al.* 1991a; 1991b; 1992) are included in the list of references that follows.

Final remark

Some readers may take this paper as negative criticism of the Deep Blue project. This would be a mistake. We have great respect for the wonderful engineering and effort and resources that has created such a powerful entity, and are grateful to IBM for carrying out the project, but, in addition, we are also deeply aware of how far AI can still go.

References

Allen, J.; Hamilton, E.; and Levinson, R. 1996. New advances in adaptive pattern-oriented chess. In *Proceedings of 8th Annual Conference on Advances in Computer Chess.*

Draper L. Kauffman, J. 1980. *Systems 1: An Introduction to Systems Thinking.* The Innovative Learning Series. Minneapolis, MN: Future Systems, Inc.

Epstein, S., and Levinson, R. 1993. *Proceedings of the AAAI Fall Symposium on Games: Planning and Learning.* Menlo Park: AAAI Press.

Gell-Mann, M. 1994. *The quark and the jaguar: adventures in the simple and the complex.* Cambridge, Massachusetts: W. H. Freeman.

Langley, P. 1996. *Elements of Machine Learning.* Morgan Kaufmann Publishers, Inc.

Levinson, R., and Snyder, R. 1991. Adaptive pattern oriented chess. In *Proceedings of AAAI-91*, 601–605. Morgan-Kaufman.

Levinson, R., and Wilkinson, J. 1997. Meta-reasoning for data analysis tool allocation. In *Proceedings of the Second Internation Symposium on Data Analysis (IDA-97).* Birkbeck College, London: University of London. To appear.

Levinson, R.; Hsu, F.; Marsland, T.; Schaeffer, J.; and Wilkins, D. 1991a. The role of chess in artificial intelligence research(panel summary). In *Proc. of IJCAI-91*, volume 2, 547–552. Morgan Kaufmann.

Levinson, R.; Hsu, F.; Marsland, T.; Schaeffer, J.; and Wilkins, D. 1991b. The role of chess in artificial intelligence research. *International Computer Chess Association Journal* 14(3):153–162.

Levinson, R.; Beach, B.; Snyder, R.; Dayan, T.; and Sohn, K. 1992. Adaptive-predictive game-playing programs. *Journal of Experimental and Theoretical AI.* To appear. Also appears as Tech Report UCSC-CRL-90-12, University of California, Santa Cruz.

Levinson, R. 1989. A self-learning pattern-oriented chess program. *International Computer Chess Association Journal* 12(4):207–215.

Levinson, R. 1992. Pattern associativity and the retrieval of semantic networks. *Computers and Mathematics with Applications* 23(6-9):573–600. Part 2 of Special Issue on Semantic Networks in Artificial Intelligence, Fritz Lehmann, editor. Also reprinted on pages 573–600 of the book, Semantic Networks in Artificial Intelligence, Fritz Lehmann, editor, Pergammon Press, 1992.

Levinson, R. 1993. Distance: Towards the unification of chess knowledge. *International Computer Chess Association Journal* 16(3):315–337.

Levinson, R. 1996. General game-playing and reinforcement learning. *Computational Intelligence* 12(1):155–176.

Omohundro, S. 1987. Efficient algorithms with neural network behavior. Technical Report UIUCDCS-R-87-1331, University of Illinois.

Russell, S., and Norvig, P. 1995. *Artificial Intelligence, A Modern Approach.* Englewood Cliffs, New Jersey: Prentice Hall.

Russell, S., and Wefald, E. 1991. *Do The Right Thing: studies in limited rationality.* Cambridge, Massachusetts: MIT Press.

Scherzer, T.; Scherzer, L.; and Tjaden, D. 1990. Learning in Bebe. In Marsland, T. A., and Schaeffer, J., eds., *Computer, Chess and Cognition.* Springer-Verlag. chapter 12, 197–216.