2005 IEEE International Conference on Systems, Man and Cybernetics
Waikoloa, Hawaii October 10-12, 2005

On the Inherent Necessity of Heuristic Proofs

Stuart H. Rubin
SPAWAR SYSTEMS CENTER
53560 Hull St.
San Diego CA 92152-5001 USA
stuart.rubin@navy.mil

James B. Law
EXPERT^2 SYSTEMS, INC.
3920 Leland St.
San Diego CA 92106-1048 USA
jamesblaw@gmail.com

Shu-Ching Chen
FLORIDA INTERNATIONAL UNIVERSITY
School of Computer Science
Miami FL 33199 USA
chens@cs.fiu.edu

Gordon K. Lee
SAN DIEGO STATE UNIVERSITY
Dept. of Electrical & Computer Engineering
San Diego, CA USA
glee@kahuna.sdsu.edu

Abstract - *It follows from the Non-Reducibility of the Theorization Problem that an arbitrary proof cannot be valid on an absolute scale. Thus, in order for an arbitrary proof to be generative, it must be self-referential; but then, it must also be heuristic if not incomplete as a consequence. By relaxing the validity requirement, heuristic (i.e., relative) proof techniques are enabled. We show that heuristics are search randomizations in space-time. It is shown how one can develop heuristics, which are randomizations of knowledge. Even more intriguing, it is shown that heuristic proof is to formal proof what fuzzy logic is to formal logic. Simply put, the paper argues for the need to relax the notion of formal proof if AI is to advance.*

Keywords: Extrema techniques, heuristics, local proofs, randomization.

1 Introduction

The theory of randomization was first published by Chaitin and Kolmogorov [1] in 1975. Their work may be seen as a consequence of Gödel's Incompleteness Theorem [2] in that it shows that were it not for essential incompleteness, then a universal knowledge base could, in principle, be constructed – one that need employ no search other than referential search. Lin and Vitter [3] proved that learning must be domain-specific to be tractable. The fundamental need for domain-specific knowledge is in keeping with the Unsolvability of the Randomization Problem [4], [5].

This paper will show that randomization reduces evolutionary time in the large. Furthermore, randomization is not recursively enumerable (r.e.) and, *a fortiori*, not recursive. Heuristics work, but the best heuristics may not be effectively discoverable, or if discovered not effectively provable.

2 Solution approach

Symmetric domains allow for the proportionate discovery of heuristics (loss or lossless randomizations). Evolution needs to utilize random and symmetric components. By maintaining the space-time definition of the processed domain as relatively random, evolutionary speed is necessarily maximized through the incorporation of heuristics. We prove that since the process of randomization is not r.e. (i.e., heuristics are incomplete – they work, but cannot be effectively covered), where search is guided by search on how to search (i.e., the evolution of heuristics is necessarily heuristic), etc., an absolute algorithm for intelligence is not provable. As a consequence, one may verify class-equivalent English pseudo-code formalizations using embedded inductive constraint proofs. For example, one might model pairs of evolving subsystems in a (closed) network configuration, which degenerate to a "coin toss". Thus, meta-rules and contextual differencing (among other heuristic methodologies) may only be *proven* using local (i.e., domain-specific) inductive randomization (e.g., in keeping with the dictates of Occam's razor), where global or domain-general proofs of correctness (or optimality) can never be had.

3 Solution methodology

3.1. Proposition

The set $\{i \mid \varphi_i : N \rightarrow N \text{ is random}\}$ is not r.e. and, *a fortiori*, not recursive.

Proof: Rubin proves that there is no effective enumeration of all the random computable functions [6] in Theorem 2.5 (*unsolvability of the randomization problem*), which appears in, "On the Auto-Randomization of Knowledge" [5].

3.2. Proposition

Let $f_0, f_1, \ldots f_i, \ldots, i \in N$, be an effective enumeration in which every f_i is a random computable function from *N* to *N*. Then, there is a random computable function $f : N \to N$, which does not appear in this effective enumeration.

Proof: To say that $f_0, f_1, \ldots f_i, \ldots, i \in N$ is an effective enumeration of random computable functions means that there is a total computable function, $g : N \to N$ such that $\varphi_{g(i)} = f$.

Define $f : N \to N$ by diagonalizing over the functions $f_0, f_1, \ldots f_i, \ldots$; that is, let

$$f(i) = f_i(i) \,||\, f_i(i) \tag{1}$$

f is defined to be a random function because $\varphi_{g(i)} = \varphi_i \,||\, \varphi_i$, where $|\varphi_{g(i)}| < |(\varphi_i \,||\, \varphi_i)|$ as a result of Theorem 3.5 (the semantic randomization problem [5]). The concatenation macro, *g*, is by appeal to Church's Thesis. For example, the concatenation of two programs, each of which adds one to its input variable results in a program that adds two to its input variable. Here, f is computable by the while-program:

```
Begin
    $X2 := g(X1);$                                   (2)
    $X1 := \Phi(X2, X1);$
End;
```

But f cannot appear in the effective enumeration $f_0, f_1, \ldots f_i, \ldots$ itself, since it would differ from a concatenation of itself on its own index. ■

3.3. Corollary. There can be no effective enumeration of all the random computable functions from *N* to *N*. This completes the proof of the proposition showing that the set $\{i \mid \varphi_i : N \to N \text{ is random}\}$ is not r.e. ■

This corollary has rather profound implications. Were the set $\{i \mid \varphi_i : N \to N \text{ is random}\}$ r.e., then one could hill-climb ever-better heuristics for solving an arbitrary computable problem. (Of course, nothing in the theory precludes the success of such an approach for a defined computable problem, given an appropriate representation of the specific domain.) However, as it turns out, the infeasibility of evolving a general heuristic solution implies that there can be no proof of such – for otherwise the proof would serve as a recursive characterization – in direct contradiction with the corollary.

It follows that all non-trivial (e.g., self-referential) proofs are inherently heuristic. That is, the notion of an absolute non-trivial and valid proof is a misnomer. Nevertheless, proofs may be partially ordered by their strength. We make use of such partial orderings to efficiently evolve heuristics of increasing complexity in Figure 2 (Section 5). The impossibility of an absolute proof in general allows transformational knowledge (e.g., a Type II KASER) to be creative [7]. Heuristic knowledge, like axiomatic knowledge in this regard, while efficacious is not provable. Next, we prove that a random set of axioms is not reducible from an arbitrary theory.

3.4 Theorem (non-reducibility of the theorization problem)

There is no algorithm, which when presented with indices *i* and *j* of an arbitrary denumerable set of axioms and a theory, $\varphi_i : s_i \to s_j$ and $\varphi_j : s_j \to s_j$ respectively, can decide whether φ_i is a randomization of φ_j. Thus, there is no algorithm, which when presented with the index *j* of an arbitrary fixed-point set (i.e., so as not to be axiomatic) $\varphi_j : s_j \to s_j$, $j \in N$, can randomize that set (i.e., transform it into φ_i, where *i* indexes a randomized set of axioms).

Proof: Define the total function

$$theorem(i, j) = \begin{cases} 1, & if\ \varphi_j = \varphi_i(\mathrm{i}); \\ 0, & otherwise. \end{cases} \tag{3}$$

$$\varphi_i(i) \downarrow \;\Leftrightarrow\; \varphi_i \ randomizes\ \varphi_j. \tag{4}$$

It is clear that $\varphi_j = \varphi_i(\mathrm{i})$ just in case $\varphi_i(i)\downarrow$. By our last observation,

$$theorem(i, j) = \begin{cases} 1, & if\ \varphi_i\ randomizes\ \varphi_j; \\ 0, & otherwise. \end{cases} \tag{5}$$

This means that the function *random*, defined in Theorem 2.5 [5], is none other than *theorem*. Since *random* is not computable [5], conclude that *theorem* cannot be either. ■

By establishing that randomization is inherently heuristic, it follows from Corollary 3.3 that the heuristics cannot be effectively enumerated as well. This can only mean that there are classes of problems (e.g., a global proof of the KASER [7]), which cannot be effectively solved. That is to say that the evolution of heuristics in general inherently depends on chance. It also must depend on symmetry for tractability. The next section makes this clear by way of example. It follows that the best way, in theory, to improve the operational KASER [7], where possible, is to construct relevant knowledge bases for self-application. That is, KASERs need be bootstrapped to efficiently evolve.

4 The heuristic 8-puzzle

Controlling the actions of a robot and automatic programming are two examples of AI applications that involve composing a sequence of operations representing the problem solution. A problem in this category is the 8-puzzle [8], which consists of an initial and goal state along with a set of operators for transforming the former into the latter. There are four operators here: Move empty space (blank) to the left, move blank up, move blank to the right, and move blank down. The 8-puzzle has a relatively small state space; there are only 9! (362,880) different configurations of the 8 tiles and the blank space.

Finding a good global database, rule set, and control strategy – the three components of a production system – is often called the *representation problem* in AI. Amarel [9] wrote a classic paper on the subject that took the reader through a series of progressively better representations for the missionaries-and-cannibals problem. Rubin [4] and Zadeh [10] provide compelling evidence that the representational formalism itself must be included in the definition of domain-specific learning if it is to be scalable.

A search algorithm is said to be *admissible* if, for any graph, it always terminates in an optimal path from a given start to a goal node whenever a path from the start to a goal node exists [8]. Heuristic power can often be gained at the expense of admissibility – i.e., power that allows for the tractable solution of more complex problems.

According to Nilsson [8], a good heuristic for the 8-puzzle is embodied by the last two terms in the equation, $f(n) = g(n) + P(n) + 3S(n)$, where $g(n)$ is the lowest cost path from a given start node to some arbitrary node found so far by the search algorithm, $P(n)$ is the sum of the distances that each tile is from "home" (ignoring intervening pieces) and $S(n)$ is a sequence score obtained by checking around the non central squares in turn, allotting 2 for every tile not followed by its proper successor and allotting 0 for every other tile; a piece in the center scores one. This heuristic function is not admissible, but allows for the rapid solution of much more difficult 8-puzzles [8]. Furthermore, non-admissible heuristics are often easier to compute, which serves to further increase their efficacy. We note that the weight in the term, $3S(n)$, serves to emphasize the heuristic component, which works well at shallow depths, but tends to fail proportionately at greater depths, where the search is necessarily more breadth-first.

The remainder of this section will address the discovery of non-admissible heuristics, scaling them with the rank of a problem (e.g., from the 8-puzzle to the 15-puzzle), and their use in transference among similar application domains (e.g., checkers and chess). Again, chance and symmetry play pivotal roles in the discovery of heuristic functions.

Figure 1 depicts the practical realization of functionals, which are theoretic objects, mapping functions to functions [6]. The transference graph presented in Figure 1 serves as a model for Minsky's, "society of mind" [11]. Here, each node can be co-evolved on a MIMD machine. The connectivity models the commissures in the brain [12]. The *Si* and *Gj* represent constraining start and goal states, respectively. Set-based operations are used in and extracted from the runs. Here, $f(n) = g(n) + [1, 3]\ Z(n)\ \mathrm{R}\ \{1, 2, 3\}\ S(n)$, where the brackets denote a range of integer values, Z denotes a class of functions, and R denotes a class of relational operators. In particular, $\{1, 3\} \rightarrow [1, 3]$ through the application of the indicated set-based generalization. Here, the search space consists of 108 symmetric functions plus random variations.

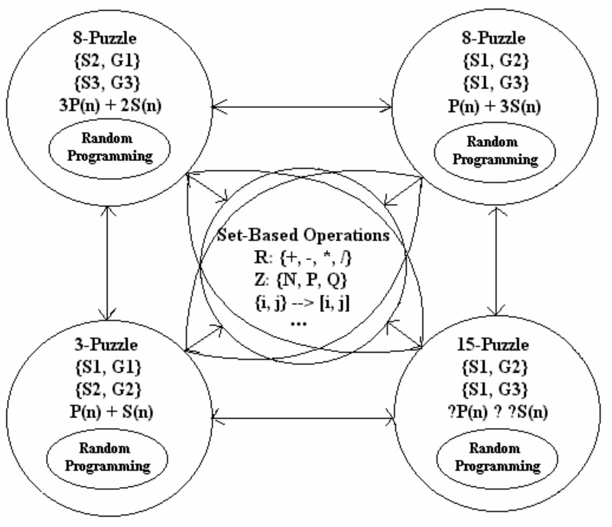


Figure 1. A society of mind

Clearly, a non-trivial domain needs to be more or less symmetric to be tractably characterized. Chance is needed to the extent that domain-specific hill-climbing yields suboptimal results.

5 Randomized local extrema

As previously explained, there is an inherent need for formal local proof techniques. Such proof techniques are predicated on more or less formal domain-specific representations, loss or lossless problem randomizations, symmetric induction, and where necessary space-time pruning, since the space of heuristic functions was shown to be non r.e. Randomized local extrema techniques enable the Type II KASER to operationally extend itself by way of self-reference.

Even local non r.e. heuristic functions can be approximately solved for using this method – given sufficient space and time. The schema for this five-step methodology is presented in Figure 2.

1. Design a domain-specific representation for the problem. Suitable problems include, but are not limited to, heuristic discovery, heuristic proofs of correctness, and heuristic optimizations. Every problem representation consists of four components; namely, a non-empty state space {S} | $\{s_o, s_1, s_2, \ldots, s_m\}$, a non-empty set of operators for moving from state to state {O} | $\{o_o, o_1, o_2, \ldots, o_n\}$, and an optional non-empty set of initial states {I} | $\{i_o, i_1, i_2, \ldots, i_p\}$ and a non-empty set of goal states {G} | $\{g_o, g_1, g_2, \ldots, g_q\}$, where $p, q \leq m$.
2. Create a random basis, which is defined to be a *min* {S, O, I, G} such that $p, q \leq m$. In particular, *min* (O) is defined to be a most general set of operators such that at least one state in *I* is mapped to at least one state in *G*, where applicable. (Note that algorithms and production rules may be placed in isomorphism, since both representations are universal.) A random basis may or may not be an abstraction of the problem as appropriate, but in either case, it needs to be a smallest sub-problem that makes sense for the domain characterization.
   REPEAT
3. Create a symmetric induction, if any, by concatenating a distinct pair of results from prior inductions (random bases), where the definition of concatenation follows from the domain-specific problem representation. More formally, {S, O, I, G} must be non-decreasing (i.e., "equal to or more-specific-than" in the context of an operator to insure convergence) in the resultant symmetric characterization.
4. Create a minimal distinct random variation of the result of a prior induction (random basis), if any. More formally, {S, O, I, G} must be non-decreasing (i.e., "equal to or more-specific-than" in the context of an operator to insure convergence) in the resultant random characterization.
   UNTIL
5. A (heuristic) solution to the problem is found, if possible, within the space-time allocation. Remove unnecessary constraints (i.e., generalize) where appropriate. Generalization is defined to be a *max* {S, O, I, G} such that $p, q \leq m$. In particular, *max* (O) is defined to be a most specific set of operators such that at least one state in *I* is mapped to at least one state in *G*, where applicable.

Figure 2. A randomized local extrema methodology

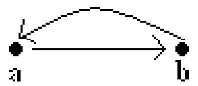
Heuristics are needed to tractably solve many different types of computable problems. Typical of the *NP-hard* problems is the *Traveling-Salesman Problem* (TSP), where the problem is to find a minimum distance tour, starting at one of several cities, visiting each city precisely once, and returning to the starting city [8]. The TSP arises in operations research [13], [14]. Let us now turn to see how our solution methodology can be successfully applied to the task of finding two heuristic solutions for the TSP. Its applicability to similar problems is by symmetric induction, while its applicability to distinct problems is by randomization. Of course, the methodology is not complete and fails if a suitable representation, randomization, or symmetric induction cannot be found (due to the *non-reducibility of the theorization problem*). A heuristic solution to the TSP is presented below.

To save space, this algorithm was presented as an object in Figure 2, where Figures 3.1 to 3.7 will instantiate it with examples – thus making clear its generalizations. It needs to be emphasized that heuristic programming differs from logic programming in that one formalizes a relaxed or fuzzy space of possibilities. For example, we may crisply write, For i = 1 to n; whereas, heuristically we may write, For i = 1 to env(n), where env(n) represents the fuzzy region of n – its environment, which of course includes n-1, n, n+1, and possibly to a lesser extent (i.e., probabilistically) the twin tails.

Example: The Traveling Salesman Problem

Step 1 (Representation). The problem generalizes to one of finding a minimum cost path over the edges of a graph {O} containing *n* nodes {S} such that the path visits each of the *n* nodes precisely once and returns to the start node {I, G}.

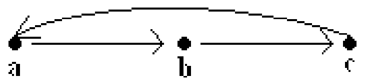
Step 2 (Random Basis). Take *n* = 2 (linear).



```
Begin
  open := {a, b}; /* min {S} */
  start := {a}; /* min {I} */
  closed := {a}; /* min {G} */
  open := open – start;
  While open <> {} do
    closed := closed ∪ member open; /* visit
    member on open */
    open := open – member open; /* min {O} */
  /* return to start is goal state */
End;
```

Figure 2.1.

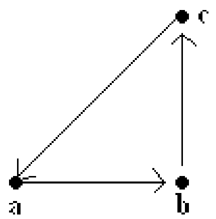
Step 3 (Symmetric Induction). Take *n* = 3 (co-linear).



```
Begin
  open := {a, b, c}; /* add "c" to set – i.e., {S} is non-
  decreasing */
  … /* same as previous randomization – {I, G, O}
  are non-decreasing */
End;
```

Figure 2.2.

Step 4 (Random Variation). Take $n = 3$ (planar).



Observe that node c has been moved up to create a right triangle. Then, the shortest path from node c to node a lies on the hypotenuse.

```
Begin
  /* same as previous randomization – {S, I, G} are
  non-decreasing */
  …
  While open <> {} do
     closed := closed ∪ nearest open; /* visit nearest
     member on open and resolve ties arbitrarily */
     open := open – nearest open; /* {O} is increasing
     because nearest is more-specific than member, of
     which it is an instance */
  /* return to start is goal state */
End;
```

Figure 2.3.

Step 5 (Evaluate and Generalize). We have progressed to the point where a first heuristic solution to the TSP can be specified in general algorithmic form. The serial complexity of this first heuristic solution can be shown to be $O(n^2)$, or $O(n^3)$ if each city is taken, in turn, for the start.

This result is none other than the familiar, "visit the nearest non-visited city next" heuristic approach for solving the TSP. A better heuristic solution (i.e., a method for finding "approximately" optimum tours) has been published by Lin [15]. Again, it is generally improper to be concerned with finding an absolute best heuristic (i.e., method for finding optimal tours), since heuristics were shown to be non r.e. Better heuristics; albeit often more complex ones, may be discovered by allocating more space-time to our solution methodology as follows. Then, assuming the evaluation in Step 5 proved to be unsatisfactory, we take the pre-generalized results of Step 4 as our last iterate:

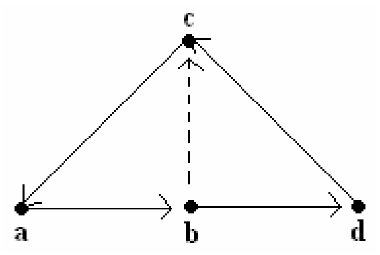
```
Begin
  open ← n nodes; /* max {S} */
  start ← arbitrarily selected node on open; /* max
  {I} */
  A closer to optimal result may be found if the
  algorithm is re-run, taking a distinct city for the start
  on each run and subsequently selecting for a
  minimal tour. Of course, this generalization will
  increase the cost of finding a solution by an order of
  magnitude.
  closed := start; /* max {G} */
  open := open – start;
  While open <> {} do
     closed := closed ∪ nearest open; /* visit nearest
     member on open and resolve ties arbitrarily */
     open := open – nearest open; /* max {O} */
  /* return to start is goal state */
End;
```

Figure 2.4.

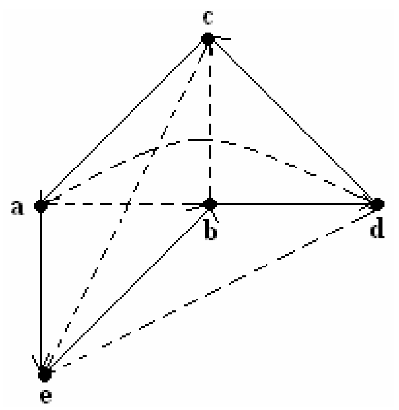
Step 3 (Symmetric Induction). Take $n = 4$ (planar).
Starting from node a, we proceed to the nearest neighbor, which is node b. Now, node c and node d are equidistant from node b, so a more-specific operator is needed to resolve the tie. It can be shown that an optimal tour here is the sequence, (a, b, d, c, a). More generally, this methodology only requires that one solve for a locally best tour.

```
Begin
  open := {a, b, c, d}; /* add "d" to set – i.e., {S} is
  non-decreasing */
  /* same as previous randomization – {I, G} are non-
  decreasing */
  …
  While open <> {} do
     closed := closed ∪ nearest2 open; /* visit
     nearest member on open, where nearest2 inherits
     all the properties of nearest, but in the event of a
     tie it selects that unvisited node (e.g., "d")
     furthest from the start. Resolve other ties
     arbitrarily. */
     open := open – nearest2 open; /* {O} is
     increasing because nearest2 is more-specific than
     nearest, of which it is an instance */
  /* return to start is goal state */
End;
```

Figure 2.5.



Step 4 (Random Variation). Take $n = 5$ (planar).



```
Begin
  open := {a, b, c, d, e}; /* add "e" to set – i.e., {S} is non-decreasing */
  /* same as previous randomization – {I, G} are non-decreasing */
  …
  Appeal to Church's Thesis [6] and define furthest to inherit all the properties of nearest2, but in the event of a tie, it selects that node on open (i.e., the set of unvisited nodes – e.g., "e") furthest from the remaining unvisited nodes (i.e., using a sum of the distances, where $\max(\sum_{j=0}^{m} x_i - x_j)$, $x_j \in open$).
  Resolve other ties arbitrarily.
  While open <> {} do
    closed := closed ∪ furthest open; /* visit furthest member on open in the event of a tie, which may be counter-intuitive */
    open := open – furthest open; /* {O} is increasing because furthest inherits from nearest2 and builds upon it using more-specific constructs */
  /* return to start is goal state */
End;
```

Figure 2.6.

Observe that node e has been added so as to create a random variation. The line segments, ab, bc, bd, and ae are all equidistant as shown. It can be demonstrated that an optimal tour here is the sequence, (a, e, b, d, c, a), where selecting e for the second city to visit always results in a better tour than if city b were selected. Previous tours are not adversely impacted by this random variation because {O} is more specific. More generally, again this methodology only requires that one solve for a locally best tour.

Step 5 (Evaluate and Generalize). We have progressed to the point where a second heuristic solution to the TSP can be specified in general algorithmic form. The serial complexity of this second heuristic solution can be shown to be $O(n^{2.x})$, or $O(n^{3.x})$ if each city is taken, in turn, for the start.

```
Begin
  open ← n nodes; /* max {S} */
  start ← arbitrarily selected node on open; /* max {I} */
  A closer to optimal result may be found if the algorithm is re-run, taking a distinct city for the start on each run and subsequently selecting for a minimal tour. Of course, this generalization will increase the cost of finding a solution by an order of magnitude.
  closed := start; /* max {G} */
  open := open – start;
  Appeal to Church's Thesis [6] and define furthest to inherit all the properties of nearest2, but in the event of a tie, it selects that node on open (i.e., the set of unvisited nodes – e.g., "e") furthest from the remaining unvisited nodes (i.e., using a sum of the distances, where $\max(\sum_{j=0}^{m} x_i - x_j)$, $x_j \in open$).
  Resolve other ties arbitrarily.
  While open <> {} do
    closed := closed ∪ furthest open; /* visit furthest member on open in the event of a tie, which may be counter-intuitive */
    open := open – furthest open; /* max {O} */
  /* return to start is goal state */
End;
```

Figure 2.7.

While the complexity of the second solution is slightly greater than that of the first solution, it will generally produce much closer to optimal solutions for larger graphs. Note that $O(n^{i}) < O(n^{i.x}) << O(n!)$, which is the complexity necessary to guarantee an optimal solution (again, since the TSP is *NP-hard*).

This is a non-trivial heuristic solution to the TSP, which may rival if not supercede the solution published by Lin [15] if his technique incorporating "repeated runs on a problem from random initial tours" is compared with our

technique, where "each city is taken, in turn, for the start". Again, better heuristics; albeit often more complex ones, may be discovered by allocating more space-time to our solution methodology.
Heuristic proofs of correctness do not require the enumeration of cases, which would otherwise be required. Heuristic proofs are useful where stronger proofs (i.e., since there can be no such thing as a valid proof of a non-trivial theorem as a corollary of the non-reducibility of the theorization problem, or [2]) are not forthcoming, or do not warrant the associated higher cost of search. A countably infinite number of proofs can only be specified in heuristic form as a consequence of the *non-reducibility of the theorization problem*. The degree to which any non-trivial theorem is inherently subject to heuristic proof has been shown to be relative – never absolute.

6 Conclusion

We note that a TSP, which requires centuries to optimize on a supercomputer can be heuristically optimized in just hours on a PC (e.g., within one percent of optimum). Similarly, we should not be surprised to find that traditional proofs, which may take centuries to find, if ever (e.g., a proof of the four-color theorem, Fermat's last theorem, *et al.*) are similarly subject to at least heuristic discovery and, with additional complexity, heuristic proof. We have endeavored to provide the reader with several examples of heuristic proof techniques. These techniques, while of potentially incredulous utility (e.g., a heuristic proof of the eight-color theorem) provide more than just another tool for the mathematician's toolbox. They show, when properly interpreted, that *Gödel's Incompleteness Theorem* [2] is an enabling, rather than a limiting result. That is, we have indirectly shown that essential incompleteness is what makes true intelligence possible.

7 Future work

Randomization in space-time has been shown to be synonymous with heuristic discovery. It remains to develop an operant mechanics for a Type II KASER and heuristically verify it using a single global fixed-point heuristic proof as supported by numerous necessarily local (heuristic) proofs.

8 Acknowledgments

The authors wish to thank Professor Lotfi Zadeh for his support and inspiration throughout the years. The authors also thank the Office of Naval Research (ONR) for financial support. This work was produced by a U.S. government employee as part of his official duties and is not subject to copyright. It is approved for public release with an unlimited distribution.

9 References

[1] G.J. Chaitin, "Randomness and Mathematical Proof," *Sci. Amer.*, Vol. 232, No. 5, pp. 47-52, 1975.

[2] V.A. Uspenskii, *Gödel's Incompleteness Theorem*, Translated from Russian. Ves Mir Publishers, Moscow, 1987.

[3] J-H. Lin and J.S. Vitter, "Complexity Results on Learning by Neural Nets," *Mach. Learn.*, Vol. 6, No. 3, pp. 211-230, 1991.

[4] S.H. Rubin, "Computing with Words," *IEEE Trans. Syst. Man, Cybern.*, Vol. 29, No. 4, pp. 518-524, 1999.

[5] S.H. Rubin, "On the Auto-Randomization of Knowledge," *Proc. IEEE Int. Conf. Info. Reuse and Integration*, Las Vegas, NV, pp. 308-313, 2004.

[6] A.J. Kfoury, R.N. Moll, and M.A. Arbib, *A Programming Approach to Computability*, Springer-Verlag Inc., New York, NY, 1982.

[7] S.H. Rubin, S.N.J. Murthy, M.H. Smith, and L. Trajkovic, "KASER: Knowledge Amplification by Structured Expert Randomization," *IEEE Transactions on Systems, Man, and Cybernetics – Part B: Cybernetics*, Vol. 34, No. 6, pp. 2317-2329, December 2004.

[8] N.J. Nilsson, *Principles of Artificial Intelligence*, Morgan Kaufmann Publishers, Inc., Mountain View, CA, 1980.

[9] S. Amarel, "On Representations of Problems of Reasoning about Actions," *Machine Intelligence*, Vol. 3, pp. 131-171, 1968.

[10] L.A. Zadeh, "From Computing with Numbers to Computing with Words – From Manipulation of Measurements to Manipulation of Perceptions," *IEEE Trans. Ckt. and Systems*, Vol. 45, No. 1, pp. 105-119, 1999.

[11] M. Minsky, *The Society of Mind*. Simon and Schuster, Inc., New York, NY, 1987.

[12] J.C. Eccles, *Understanding of the Brain*, McGraw-Hill Co., 2d ed., New York, NY,1976.

[13] H. Wagner, *Principles of Operations Research* (2d ed.), Prentice-Hall, Inc., Englewood Cliffs, NJ, 1975.

[14] F.S. Hillier and G.J. Lieberman, *Introduction to Operations Research* (2d ed.), Holden Day Publishers, Inc., San Francisco, CA, 1974.

[15] S. Lin, "Computer Solutions of the Traveling Salesman Problem," *Bell System Tech. Journal*, Vol. XLIV, No. 10, pp. 2245-2269, 1965.