Structuring and Discovery: Artificial Intelligence and Electronic Commerce[*]

Steven O. Kimbrough
University of Pennsylvania
565 Jon M. Huntsman Hall
3730 Walnut Street
Philadelphia, PA 19104-6340
kimbrough@wharton.upenn.edu
215-898-5133

PAIW talk, 25 April 2003, http://www.ai.uga.edu/paiw2003.html

[*]File: paiw-nute-foils.pdf.

Context & Framing

- Encoding problem:

  How can knowledge effectively and usefully be <u>encoded</u> automatically in machines, that they may apply it for our purposes?

  Example technique: neural nets. Note: supervised learning.

- Structuring problem:

  How can knowledge be <u>structured</u> so that it can effectively and usefully be processed automatically in machines, that they may apply it for our purposes?

  Example technique: expert systems.

Context & Framing

- Discovery problem:

  How can knowledge be discovered automatically by machines, that we may apply it for our purposes?

  Example technique: genetic algorithms. Note: unsupervised learning; autonomous agents.

Plenty to talk about vis à vis electronic commerce. Will focus on structuring and discovery.

Goals and Plan

- A sampling of developments, promising ideas, trends, etc. in the intersection of AI and e-commerce.

  Interplay of ideas, topics, and examples.

- Frame these ideas and provide a larger map.

- Begin with a bit more framing and overview.

- Work with a broad concept of electronic commerce.

- In the interests of informativeness, will avoid for the most part obvious e-commerce topics, e.g., personalization at Web sites.

Topics

- Exploiting weak Structure

  Text mining. Semi-structured data.

- Creating & exploiting structure

  Data mining. Sublanguages. ebXML.

- Discovery

  Metaheuristics. Artificial agents.

Exploiting Weak Structure:

Text Mining

PageRank™

1. Underlies Google
2. How does it work?
3. Lession #1: There are clever ways to extract information automatically from weakly-structured sources.
4. Lesson #2: (later)

PageRank Explained

PageRank relies on the uniquely democratic nature of the web by using its vast link structure as an indicator of an individual page's value. In essence, Google interprets a link from page A to page B as a vote, by page A, for page B. But, Google looks at more than the sheer volume of votes, or links a page receives; it also analyzes the page that casts the vote. Votes cast by pages that are themselves "important" weigh more heavily and help to make other pages "important."

Important, high-quality sites receive a higher PageRank, which Google remembers each time it conducts a search. Of course, important pages mean nothing to you if they don't match your query. So, Google combines PageRank with sophisticated text-matching techniques to find pages that are both important and relevant to your search. Google goes far beyond the number of times a term appears on a page and examines all aspects of the page's content (and the content of the pages linking to it) to determine if it's a good match for your query.

[From: http://www.google.com/technology/]

Exploiting Weak Structure: The Plato Problem

Imagine that you have access to a large document collection and are interested in obtaining information about Plato, the ancient Greek philosopher. You successfully retrieve all documents containing the word 'Plato'. When you examine these documents you find what you expected: many of them are indeed about the Greek philosopher; others are about various commercial products trading on the philosopher's name, but having nothing to do with him.

You are then surprised when a friend presents you with a completely different set of documents, also retrieved from the same large document collection. Nearly all of these new documents are very much about Plato the philosopher, yet none of them mention 'Plato' (or any close term, e.g., 'Platon' a foreign spelling).

How can this be? Your friend explains. 'Plato' was not the philosopher's orginal name. It was a nickname given to Aristocles by his wrestling coach ('personal trainer' in today's language) and it means 'chubby' or 'chubs'. Somehow it stuck. Many authors, however, have written about the philosopher using only his orginal name. These writings would not be retrieved by a keyword search using 'Plato'.

The Plato Problem is the problem of designing a computer system that helps in finding relevant and useful documents that do not contain the search term(s) you have in mind. The scope of this problem extends far beyond the world of classical scholarship. In fact, except in relatively trivial cases, it appears much more often than not.

From the STAIRS study (Blair & Maron)

Sometimes we followed a trail of linguistic creativity through the database. In searching for documents discussing "trap correction" (one of the key phrases), we discovered that relevant, unretrieved documents had discussed the same issue but referred to it as the "wire warp." Continuing our search, we found that in still other documents trap correction was referred to in a third and novel way: the "shunt correction system." Finally, we discovered the inventor of this system was a man named "Coxwell" which directed us to some documents he had authored, only he referred to the system as the "Roman circle method." Using the Roman circle method in a query directed us to still more relevant but unretrieved documents, but this was not the end either. Further searching revealed that the system had been tested in another city, and all documents germane to those tests referred to the sysem as the "air truck." At this point the search ended, having consumed over an entire 40-hour week of on-line searching, but there is no reason to believe that we had reached the end of the trail; we simply ran out of time.

The Plato Problem: Computational (AI) response?

- Intuition: Relevant documents <u>with</u> the term you search on look rather like relevant documents <u>without</u> the search term.
- Find a profile of terms for the relevant documents having the search term, then find similar documents without the term.
- Such algorithms can rank collections by relevance.
- Expensive. Good performance.

The Raynaud Problem

You have a daughter who has been diagnosed with an unpleasant disease called Raynaud's Syndrome. You are told that there is no known effective treatment of the disease, and you quickly verify this by a thorough search of all the on-line medical research papers.

Concerned about your daughter, you wonder if, somehow, a solution has been found, but no one has "made the connection" with Raynaud's Syndrome. The Raynaud Problem is the problem of designing a computer system that will help you in following up on your hunches, specifically here and in general. Can you find a plausible hypothesis for a treatment in the existing literature?

The Raynaud Problem: Computational response?

- Originally, Don Swanson of U. Chicago
- Intuition: there may be undiscovered knowledge implicit in document collections
- Idea: For topic A, find documents on topic B, mentioned by the documents of type A, and find documents on topic C, where the C documents mention the B documents, but the A and C documents don't mention each other. Hypothesize an undiscovered association between topic C and topic A, via topic B.

The Raynaud Problem: Computational response?

- Example: A=documents on Raynaud's. B = documents on blood viscosity. C= documents on fish oil.
- This and other hypotheses confirmed in clinical studies
- Manual discovery subsequently supported with automation and computationally duplicated

Lesson #2 from PageRank:

- Exploit the work of others.

Note: Swanson's work preceded, and was published before, PageRank.

The Practice Problem

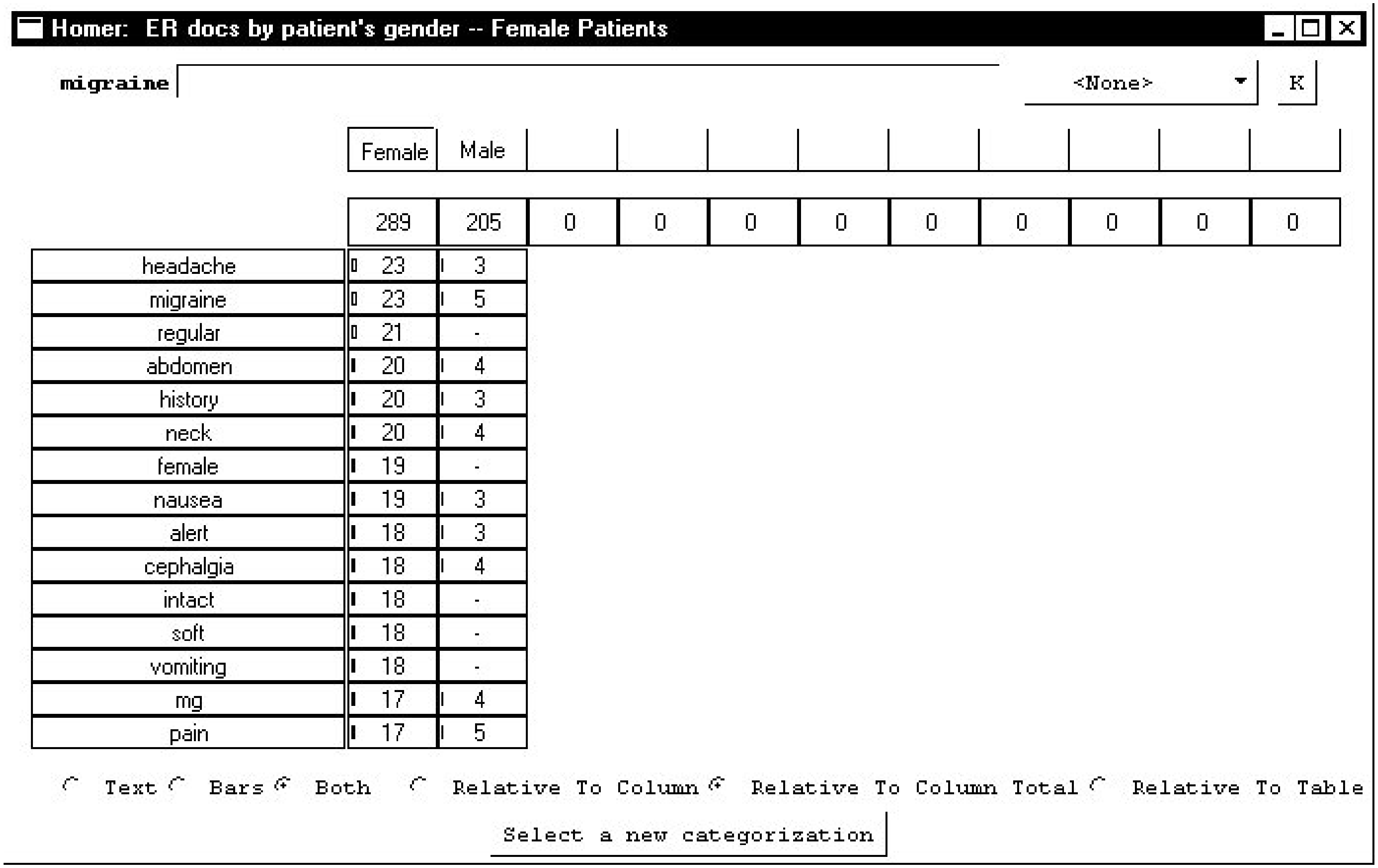
You are a hospital administrator and you would like to have information about how the practices of your emergency room doctors differ. Do they treat similar patients differently? If so, how are the treatments different? Do they see different kinds of patients? and so on. Unfortunately, the only electronic records you have of the patient visits are in document form. There is no database to mine, just a series of emergency room documents.

The Practice Problem is the problem of designing a computer system that will help you find and see meaningful patterns of behavior, based on information extracted from weakly-structured documents. These patterns of behavior can only emerge from the document collection as a whole. The information is not present in any small number of documents. It is implicit in the collection, not explicit in any document.

Garett Dworman: Homer

Patterns in text collections. Use of term-term co-occurrence data extracted from text. Experimental support for efficacy, e.g., with emergency room records in XML.

`http://homepage.mac.com/garett_dworman.`
`http://homepage.mac.com/garett_dworman/homer/HomerER.html`
`http://homepage.mac.com/garett_dworman/homer/`
`HomerLaughlinArchitecture.html`
`http://homepage.mac.com/garett_dworman/homer/`
`HomerLaughlinFantasy.html`

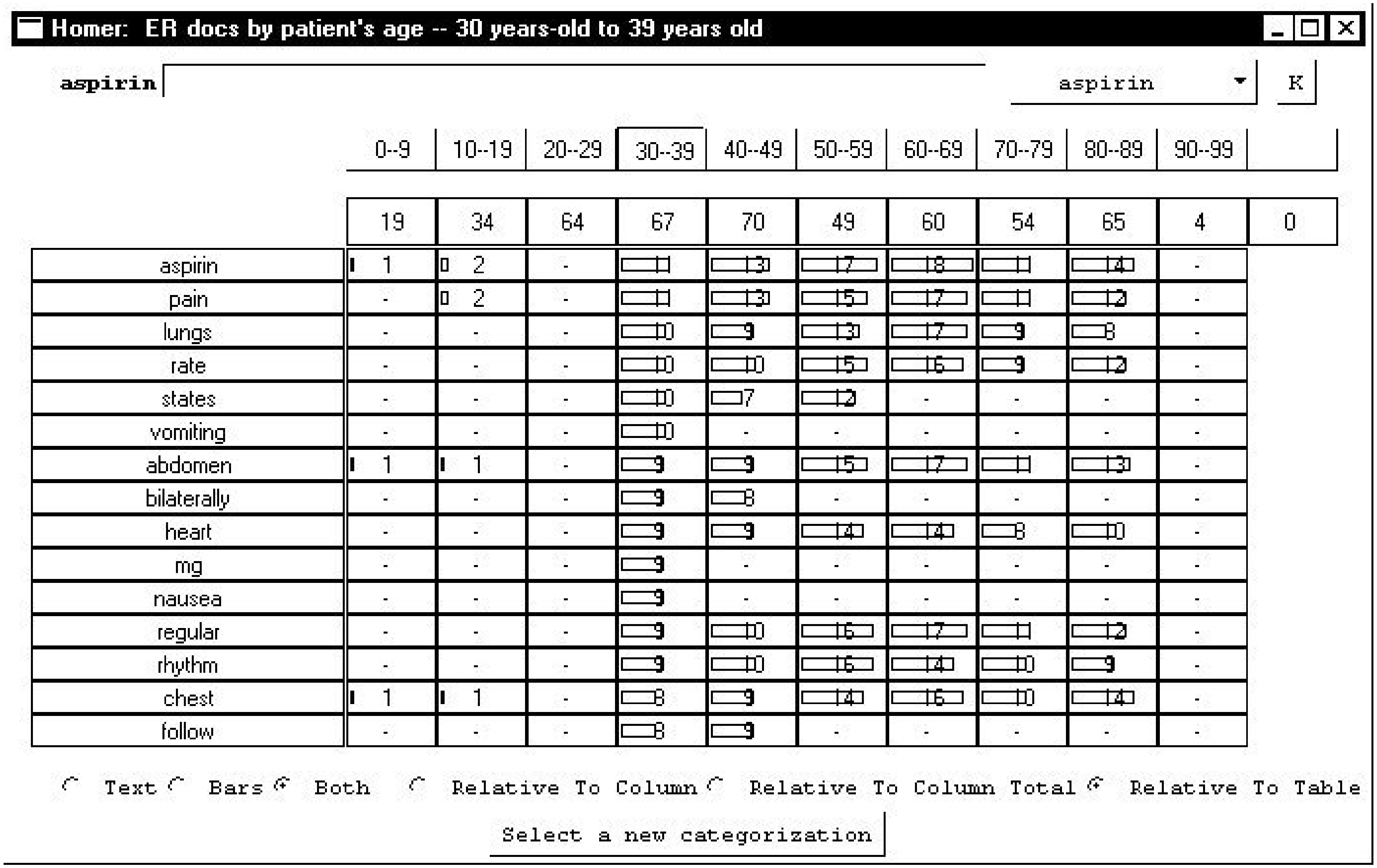

Homer: ER docs by patient's gender -- Female Patients

migraine

	Female	Male									
	289	205	0	0	0	0	0	0	0	0	0
headache	23	3									
migraine	23	5									
regular	21	-									
abdomen	20	4									
history	20	3									
neck	20	4									
female	19	-									
nausea	19	3									
alert	18	3									
cephalgia	18	4									
intact	18	-									
soft	18	-									
vomiting	18	-									
mg	17	4									
pain	17	5									

Text Bars [✓ Both] Relative To Column [✓ Relative To Column Total] Relative To Table

Figure 1: Pattern of Migraine Headaches



Homer: ER docs by patient's age -- 30 years-old to 39 years old

aspirin

aspirin

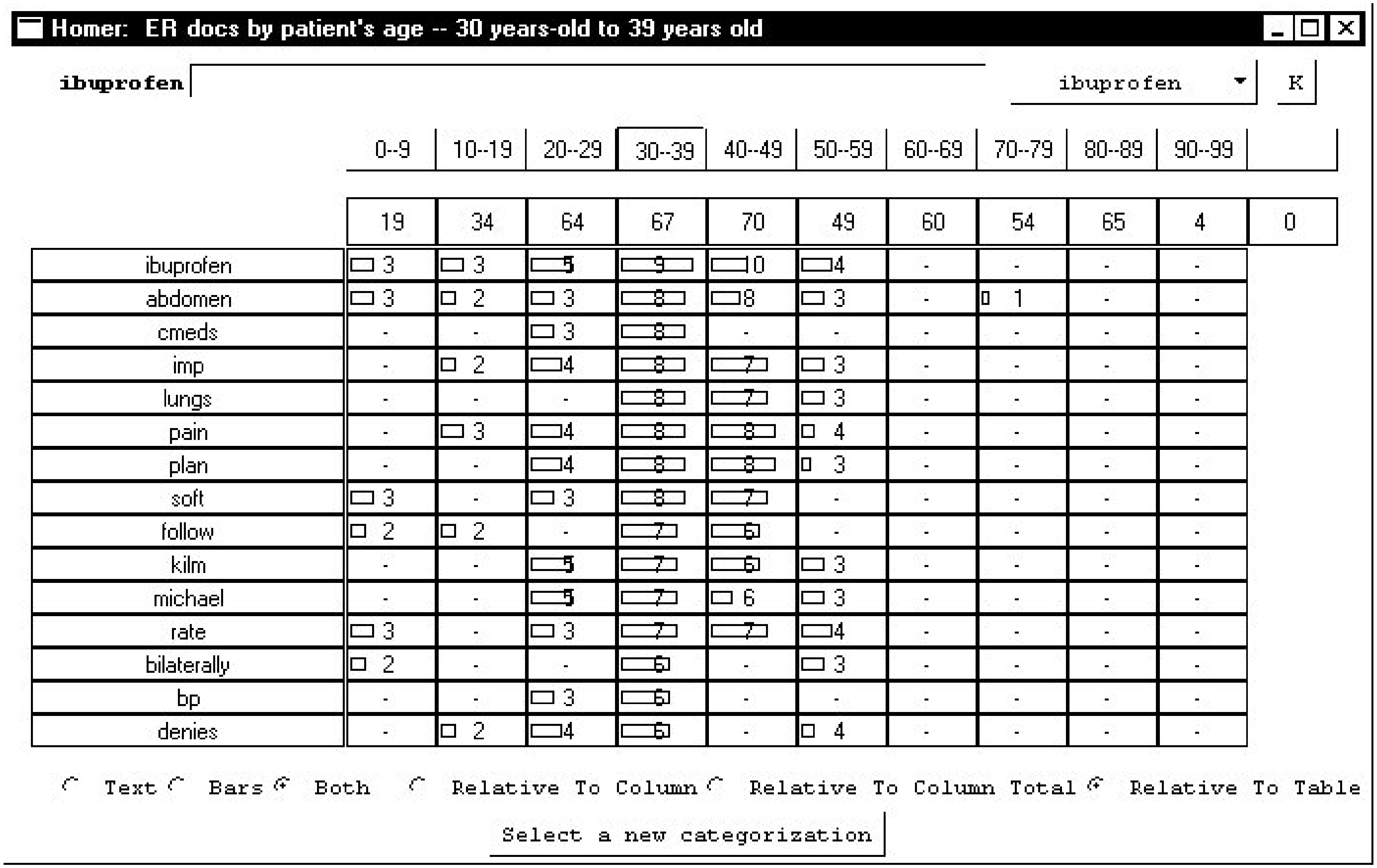
K

	0--9	10--19	20--29	30--39	40--49	50--59	60--69	70--79	80--89	90--99	
	19	34	64	67	70	49	60	54	65	4	0
aspirin	1	2	-	11	13	17	18	11	14	-	
pain	-	2	-	11	13	15	17	11	12	-	
lungs	-	-	-	10	9	13	17	9	8	-	
rate	-	-	-	10	10	15	16	9	12	-	
states	-	-	-	10	7	12	-	-	-	-	
vomiting	-	-	-	10	-	-	-	-	-	-	
abdomen	1	1	-	9	9	15	17	11	13	-	
bilaterally	-	-	-	9	8	-	-	-	-	-	
heart	-	-	-	9	9	14	14	8	10	-	
mg	-	-	-	9	-	-	-	-	-	-	
nausea	-	-	-	9	-	-	-	-	-	-	
regular	-	-	-	9	10	16	17	11	12	-	
rhythm	-	-	-	9	10	16	14	10	9	-	
chest	1	1	-	8	9	14	16	10	14	-	
follow	-	-	-	8	9	-	-	-	-	-	

Text Bars [✓ Both] Relative To Column Relative To Column Total [✓ Relative To Table]

Select a new categorization

Figure 2: Pattern of Aspirin Use



Homer: ER docs by patient's age -- 30 years-old to 39 years old

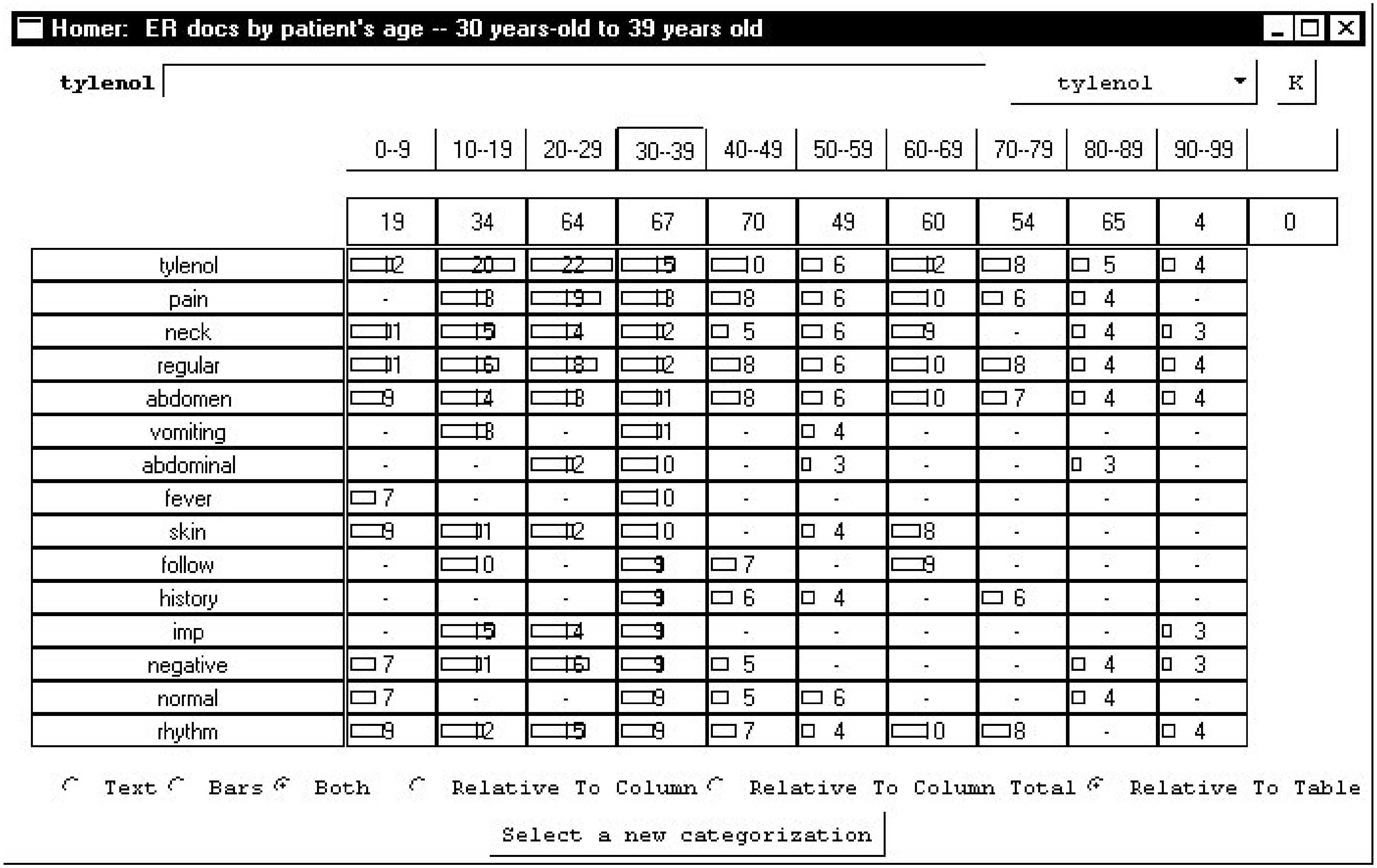
ibuprofen | ibuprofen | K

	0--9	10--19	20--29	30--39	40--49	50--59	60--69	70--79	80--89	90--99	
	19	34	64	67	70	49	60	54	65	4	0
ibuprofen	3	3	5	9	10	4	-	-	-	-	
abdomen	3	2	3	8	8	3	-	1	-	-	
cmeds	-	-	3	8	-	-	-	-	-	-	
imp	-	2	4	8	7	3	-	-	-	-	
lungs	-	-	-	8	7	3	-	-	-	-	
pain	-	3	4	8	8	4	-	-	-	-	
plan	-	-	4	8	8	3	-	-	-	-	
soft	3	-	3	8	7	-	-	-	-	-	
follow	2	2	-	7	6	-	-	-	-	-	
kilm	-	-	5	7	6	3	-	-	-	-	
michael	-	-	5	7	6	3	-	-	-	-	
rate	3	-	3	7	7	4	-	-	-	-	
bilaterally	2	-	-	6	-	3	-	-	-	-	
bp	-	-	3	6	-	-	-	-	-	-	
denies	-	2	4	6	-	4	-	-	-	-	

Text Bars [✓ Both] Relative To Column Relative To Column Total [✓ Relative To Table]

Select a new categorization

Figure 3: Pattern of Ibuprofen Use



Homer: ER docs by patient's age -- 30 years-old to 39 years old

tylenol

tylenol ▾ K

	0--9	10--19	20--29	30--39	40--49	50--59	60--69	70--79	80--89	90--99	
	19	34	64	67	70	49	60	54	65	4	0
tylenol	12	20	22	15	10	6	12	8	5	4	
pain	-	18	19	18	8	6	10	6	4	-	
neck	11	15	14	12	5	6	9	-	4	3	
regular	11	16	18	12	8	6	10	8	4	4	
abdomen	9	14	18	11	8	6	10	7	4	4	
vomiting	-	18	-	11	-	4	-	-	-	-	
abdominal	-	-	12	10	-	3	-	-	3	-	
fever	7	-	-	10	-	-	-	-	-	-	
skin	9	11	12	10	-	4	8	-	-	-	
follow	-	10	-	9	7	-	9	-	-	-	
history	-	-	-	9	6	4	-	6	-	-	
imp	-	15	14	9	-	-	-	-	-	3	
negative	7	11	16	9	5	-	-	-	4	3	
normal	7	-	-	9	5	6	-	-	4	-	
rhythm	9	12	15	9	7	4	10	8	-	4	

Text Bars [✓ Both] Relative To Column Relative To Column Total [✓ Relative To Table]

Select a new categorization

Figure 4: Pattern of Tylenol Use

The Anonymous Problem

Your are presented with a text—a sample of speech, a document or a portion of a document—whose authorship is unknown or in doubt. Can you identify the author or must the text remain anonymous? Can the date or influencing circumstances of the text be identified?

- Computational response

  Count words, frequencies, patterns. Many successes.

The Product Placing (P^2) Problem

Consider the prototypical case of a firm having the rights to C, an intellectual property (IP). Think of C as a component product, rather than an end product, E. C is neither shoes, nor ships, nor sealing wax. Instead, C may be a valuable component (e.g., substance or process) in making shoes, or ships, or sealing wax, etc. The *product placing* (P^2) problem has these main aspects:

1. For which end products, the Es, is C a promising component?
2. Who, including possible partners, is well-positioned to commercialize the Es, using C?

In addition, consider an end product, E, or a problem, P. Think of P as a recognized shortcoming in a recognized end product. P is too much labor content, too little resistance to sunlight, unwanted reliance on nonrenewable resources, and so on. (Cost reduction is always and available P.) Thus, the product placing problem also has these main aspects:

3. For a given end product, E, find previously unrecognized components, Cs, with promise of improving E.

4. For a given problem, P, find previously unrecognized end products, E, using component products C, that provide an improved solution to the problem.

5. Who, including possible partners, is well-positioned to commercialize such improvements to product E and solutions to problem P?

The Product Placing (P^2) Problem: Computational Response?

- Recall: Lesson #2 from PageRank: Exploit work by others
- Classification schemes: a new source of work to be exploited
- Intuition: Search on the terms describing the properties of C, and find relevant documents. Classify the documents (the hits) using a pertinent classification scheme. Identify promising topics or areas by the density of hits.

The Product Placing (P^2) Problem: Computational Response?

- Example: 1,3-propanediol (PDO), and the USPTO classification scheme
- At random, idea number 66: Personal Care/Cosmetics.

  Component in water-absorbent resin composition. Water-absorbent resins absorb body of liquids and are widely used as components of sanitary materials such as disposable diapers, sanitary napkins and incontinence pads. PDO is a suitable surface-cross-linking agent, which enhances the water absorption properties of the composition.

Exploiting Weak Structure:

Semi-Structured Data

Semi-Structured Data

- DBMS $\approx$ structured data
- free text $\approx$ unstructured data
- marked-up text $\approx$ semi-structured data

  HTML. SGML. XML.
- SQL for XML? Yes.
- How can semi-structured data be exploited?

Semi-Structured Data

1. *Building specific data mining models.* This can help develop strategies for handling new situations that arise, for finding profiles of problematic situations, for detecting successful and unsuccessful policies, and so on. For example, given a new case or situation, elements from the case as it unfolds can be used to retrieve similar cases. Among these cases, various classification models, such as C4.5 can be built to distinguish conditions that result in a favorable outcome with those that result in unfavorable outcomes. For example, the classification model built may show that if there is a specific set of actions taken, then favorable outcomes usually result. This is clearly operationalizeable.

Semi-Structured Data

2. *Learning interesting patterns.* It would be interesting if we find that even when the recommended actions are taken, if the case is a medical emergency in the North Atlantic, then bad outcomes tend to result. This is an example of a pattern in data, and there has been much work done in techniques that can learn “interesting patterns from data. These patterns may be pieces of potentially useful information, and are different from classification models in that they are more general. Other examples of interesting patterns may be those that suggest novel lines of legal reasoning.

Semi-Structured Data

3. *Querying for explicit items.* Here is a partial list of potential queries that would be possible were fully formalized or semistructured corpora available in scale and in scope.

1. We could trace the evolution of a legal concept by calculating the transitive closure of references to the concept (in other words, the grammar should include explicit formalisms for the nature of references - like specialization, generalization, combination, subdivision - see UML) For example, applying the 1921 Martin Act, a broad, New York State law barring fraud in the sale or offering of securities. Eliot Spitzer, New York Atty General is using this to prosecute and regulate Wall Street.

2. We could check the quality and consistency of documents by defining

the concepts of well-formedness and validity with respect to the lexicon and grammar.

3. We could check the quality and consistency by defining additional constraints (inclusion, inverse, domain, etc.).

4. We could support robust, flexible querying through a corpus of decisions and legal documents.

5. We could exploit active database concepts (event-condition-action) rules to automatically update indexes to support robust querying.

6. We could automatically update references among legal decisions and policy documents.

7. Given a well-defined relationship between concepts (see below), we could

search explicitly for novel instances of a concept within a legal domain for novel lines of legal reasoning.

8. We could search for relationships within dissimilar legal domains for parallel relationships to uncover novel lines of legal reasoning.

Creating & Exploiting Structure

Exploiting Structure: Data Mining

- Matured into regular practice. Continuing innovation and development.
- Two problems:
  1. Finding patterns, or rules, in the data.
     (Presence of noise, incompleteness, etc.)
  2. Finding <u>interesting</u> patterns in the data.

Data Mining: Computational Response?

- Finding patterns. Emphasis on machine learning algorithms, broadly construed. Will discuss algorithms later under the heading of *metaheuristics*.

- Finding interesting patterns.

  Usually, easy to find lots of 'significant' patterns, only a few of which are surprising or interesting in any way. What to do? Intuition: Suppose $A \rightsquigarrow Z$ (unsurprisingly) but you find $A \wedge B \rightsquigarrow \neg Z$ (surprisingly!). B here is a sort of *defeater* as in defeasible reasoning. Excellent results achieved in this way (understood generally) by Balaji Padmanabhan.

Formalizing Knowledge

- Aka: knowledge engineering, an expensive pursuit.
- Consider how you would extract knowledge from a corpus of documents (e.g., legal documents or records):
  1. Manually symbolize the corpus
  2. Accept minimally structured documents
  3. Automatically structure the corpus
- Option #4: Create the records originally in structured form.

How Might/Are Records/Documents Be Originated in Semi/Structured Form?

- Semi/Structured form: Design protocols and representation schemes for messages and/or records. Usage then creates semi/structured data originally.

  Hugely important example: ebXML:http://www.ebxml.org/; OASIS: http://www.oasis-open.org/

- (Potentially) fully structured form: Design <u>sublanguages</u> for special purposes, and formalize them.

  Example: Seaspeak.

- Also, design special database representations for accommodating special sublanguages, including, e.g., representation of obligations and permissions, defeasibility, speech acts, &c. Notable work done by Alan Abrahams.

Discovery:

Metaheuristics

Metaheuristics

- Explosive interest and productivity ongoing.
- Convergence of AI and Management Science.
- heuristic $\approx$ a solution to a problem (or a strategy in a game), not necesssarily optimal ('rational'), but hoped to be warranted
- metaheuristic $\approx$ a procedure for generating heuristics
- Two kinds of metaheuristics:
    1. Population-based (evolution programs)
    2. Local-search-based

Appropriate Problems: Searching for (optimal) combinations

Very many problems involve fundamentally a search for happy combinations of basic elements.

- Loading boxes on a truck, airplane, or ship.
- Scheduling activities, or resources in a factory.
- Assigning people to jobs.
- Composing a portfolio of equities.
- Finding a better protein by varying the constituent amino acids

And much more. (What isn't a combinatorial search problem?)

Think: letters into words, words into sentences.

Combinatorial Search Is Hard

- 10,000 monkeys?
- Astronomical is *small* compared to common search problems. About 10^{80} atomic particles in the universe.
- $2^{1024} \approx 10^{300}$ possible icons, 32×32 bits (no color).
- $20^{1000} = 2^{1000} \times 10^{1000}$ proteins of length 1000 (smallish).
- $2^{500} \approx 10^{150}$ distinct portfolios of 500 stocks, neglecting how much of each equity is in the portfolio.

- Permutations: factorial $(n(n-1)(n-2)\ldots(2)(1))$ grows very quickly. Orderings, rankings of things.

  `=fact(100)` $\rightsquigarrow$ 9.3326E+157

```
60 secs/min
60 min/hour
24 hrs/day
366 days/year
 31,622,400  sec/year
15,000,000,000 years since Big Bang
 474,336,000,000,000,000  secs since Big Bang
```

  If we examined a trillion permutations per second, we could do $5 \times 10^{17+12} < 10^{30}$ cases since the beginning of the world. This would hardly make a dent at all in $100!$ permutations. $10^{157}/10^{30} = 10^{127}$.

What to Do?

- How can we possibly approach such problems? Must we despair?
- Learn from Mother Nature
- Shortly, but first. . .

A Class of Characterizing Problems

1. *Recognition & specification of solutions.* You can recognize, and can specify the form of, a solution, of any valid solution, in a precise way.

   Examples: Loading items on an airplane: a solution consists of a list of the items to be placed on the plane. That's precise. It may or may not be a good solution, or even feasible, but every solution can be described in this way.

2. *Measurement of solution quality.* You have a way of measuring the quality of every possible solution.

   Some solutions may be infeasible, but you need some quantitative scoring mechanism to distinguish the better from the worse. Example: in an optimization problem, the value of the objective function for a feasible solution is the measure.

A Class of Characterizing Problems (con't.)

3. *System representation.* You can model, or represent, the system of interest.

   Since the systems you are typically interested in are not inherently computational—usually they are physical systems such as real markets, proteins, production lines, etc.—you will need an acceptably accurate model of the system of interest.

4. *Solution generation.* You have a reasonably efficient and effective way to create solutions, which may subsequently be evaluated for their qualities.

   Since you typically work with a model, you'll have to generate solutions to that representation.

5. *Reasonably smooth variation of quality in solutions.* Not just a needle in a haystack. Generally, different solutions have different values (fitness values).

Many examples, including

- Function maximization (minimization)

  Linear programming, and constrained optimization in general. A solution is any valid instance of the objective function, any setting of its variables.

- Rule finding or discovery; prediction

  Data mining. Robotics and control. Finance applications.

- Finding a path between two places or states

  How to get there from here.

- Finding strategies for playing games

Combinatorial Search

- In general: discovering useful combinations of things.

  Recall: things can be mapped to, represented by, 1s and 0s.

- Let's call these *combinatorial search problems*.

- The idea is that there's a solution space, which it makes sense to search through.

- We have seen that the search spaces can easily be beyond huge, beyond astronomical.

Evolution Programs

1. Initialization: set parameters and policies for the run; generate (and maintain) a population of (candidate) solutions.
2. Evaluate (determine fitness of) members of the population.
3. Modify the constitution of the population based on: (a) the evaluation (step 2), (b) various operators, and (c) the parameters and policies of the run.
4. Iterate the process by going to step 2, and continuing until a stopping condition is reached.

Figure 5: Algorithm Outline for Evolution Programs

Evolution Programs: Examples

- genetic algorithms
- genetic programming
- evolution strategies
- evolutionary programming
- artificial immune systems (AIS)
- memetic algorithms (e.g., particle swarm optimization)
- reinforcement learning: LCS: Learning Classifier Systems

Local Search Metaheuristics

1. Initialization: set parameters and policies for the run, $\vec{p}$.
2. Create a new case by creating a new (candidate) solution, s.
3. Apply variation operators to s, yielding a new candidate solution, s'.
4. If $f(s', \vec{p})$ improves $f(s, \vec{p})$, set $s \longleftarrow s'$.
   Note: $f(x, \vec{y})$ is a problem-specific fitness or evaluation function for the candidate solution x in the presence of run parameters $\vec{y}$; it need not be deterministic.
5. Update $\vec{p}$
6. If the case is not done, go to step 3
7. If run is not done, go to step 2

Figure 6: Algorithm Outline for Local Search Heuristics

Local Search: Examples

- simple hillclimbing
- simulated annealing
- neural networks
- tabu search
- demon algorithms
- extremal optimization
- scatter search
- reinforcement learning: TD-learning, Q-learning, etc.

A New Addition: Two-Population GA

- Combinatorial search with constraints poses serious problems for GAs (and EPs generally).
- How to handle the constraints, since the GA operators will very often create infeasible children?
- Natural move: penalty functions. But these will mislead if the penalties are not well-chosen (and there are no guidelines).
- Alternative approach: Use a two-population GA. . .

Sample Problem Sets

Problem	Min/Max	Objective	# Variables	# Linear Inequalities	# Nonlinear Inequalities
11 (test11)	max	Nonlinear	2	0	2
12 (chance)	min	Linear	3	2	1
13 (circle)	min	Linear	3	0	10
14 (ex3_1_4)	min	Linear	3	2	1
15 (ex7_3_1)	min	Linear	4	6	1
16 (ex7_3_2)	min	Linear	4	6	1
17 (ex14_1_1)	min	Linear	3	0	4
18 (st_e08)	min	Linear	2	0	2
19 (st_e12)	min	Nonlinear	3	3	0
20 (st_e19)	min	Polynomial	2	1	1
21 (st_e41)	min	Nonlinear	4	0	2

Table 1: Summary of Group 2 Problems

Representative Results

Problem	Best Known or Optimal*	Genocop III			Two-Population GA		
		Best of 10	Median	Std.	Best of 10	Median	Std.
11	?	0.115047	0.11504	4.87E-05	0.115047	0.115047	4.16E-08
12	29.8943781591	29.89549	29.94807	0.034976	†	29.91486	0.032017
13	4.57424778502	4.574318	4.575747	0.027615	4.574248	4.574257	6.67E-05
14	-4.0000	-4	-4	0.032482	-4	-3.99991	0.000131
15	0.341739553124	0.3558	0.416292	0.141607	‡	0.320127	0.00033
16	1.08986397147	1.09145	1.113644	0.033588	1.089952	1.089994	2.56E-05
17	0.0000	1.44E-10	1.19E-06	2.85E-06	4.69E-05	6.63E-05	1.36E-05
18	?	0.741782	0.741782	9.93E-09	0.741782	0.741782	5.58E-08
19	?	-4.5099	-4.49564	0.02252	-4.51347	-4.51319	0.000995
20	?	-118.705	-118.704	0.021621	-118.705	-118.705	0.000187
21	?	645.626	648.9749	4.398663	641.8244	641.8283	2.663347

Table 2: Summary of Group 2 Results for a Two-Population GA. † = 29.8943786178232, ‡ =**0.319777729979837**.

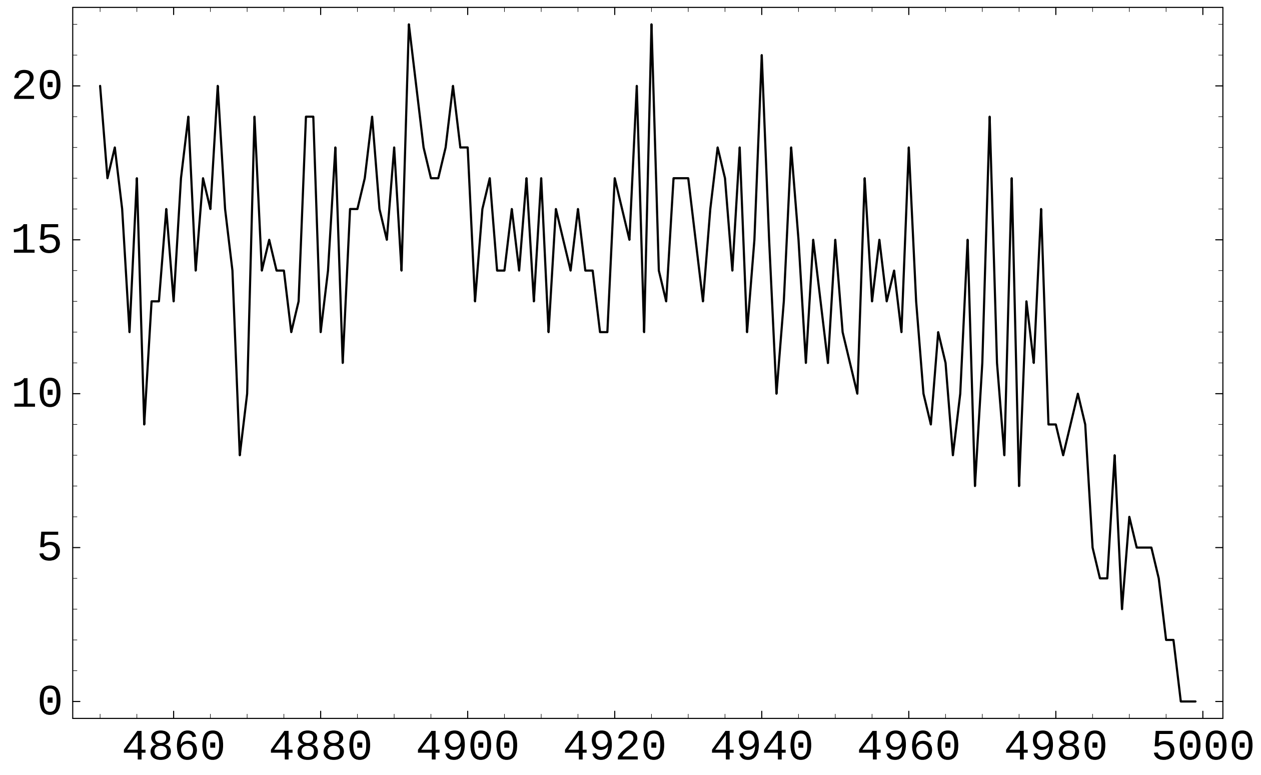


Figure 7: For problem ex7_3_2 with population size 50: count of number of infeasible solutions created from the feasible population, by generation, 4850–4999.

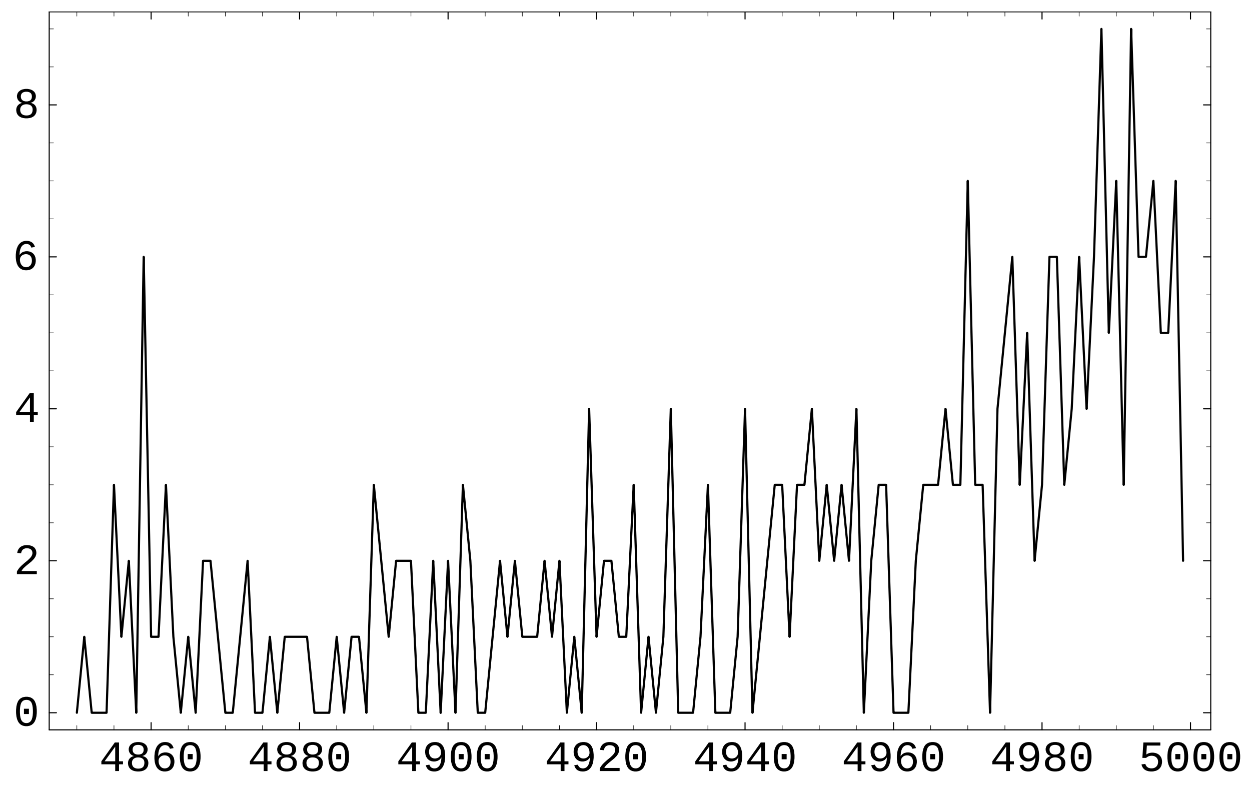


Figure 8: For problem ex7_3_2 with population size 50: count of number of feasible solutions created from the infeasible population, by generation, 4850–4999.

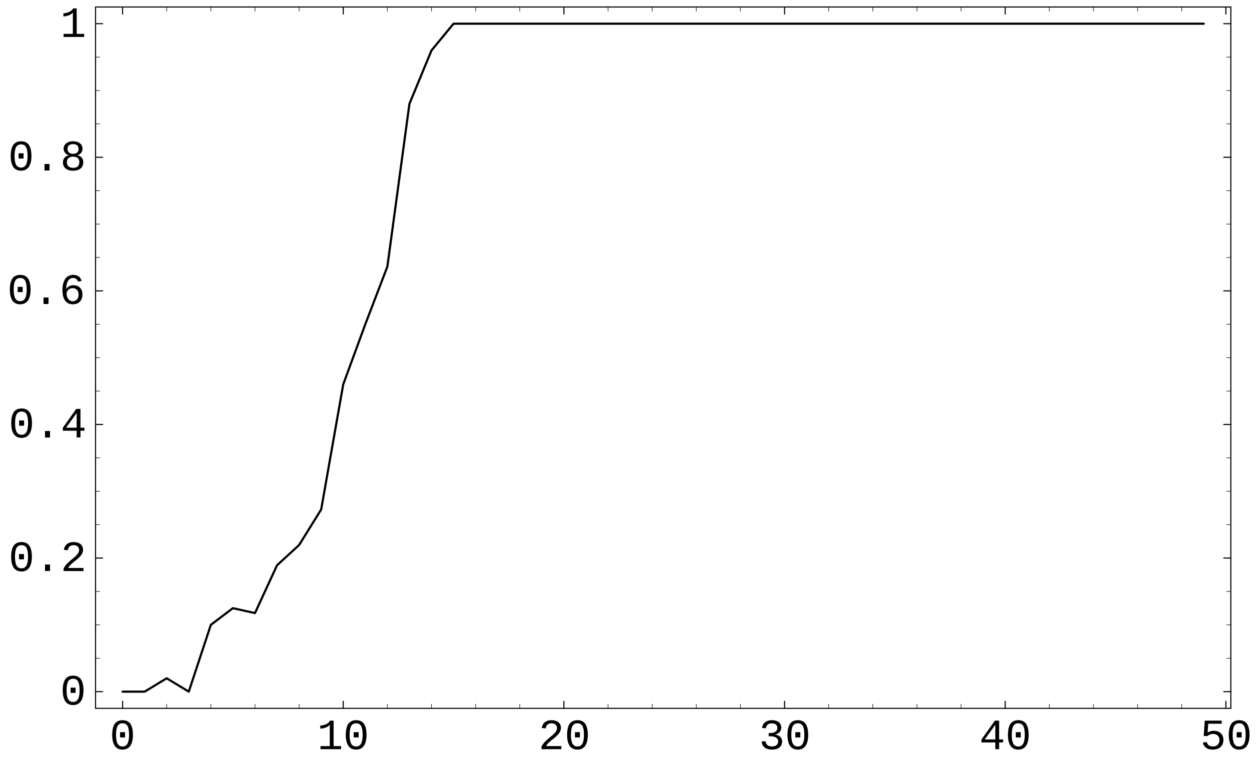


Figure 9: For problem ex7_3_2 with population size 50: fraction, by generation, of feasible individuals with an infeasible ancestor.

Discovery:

Artificial Agents

Game of Life Sites

1. www.math.com/students/wonders/life/life.html

   A nice Java applet with supporting info.

2. www.bitstorm.org/gameoflife/

   A Java applet for Life.

3. psoup.math.wisc.edu/Life32.html

   Life32. A very nice program for the Wintel environment.

4. www.mirwoj.opus.chelm.pl

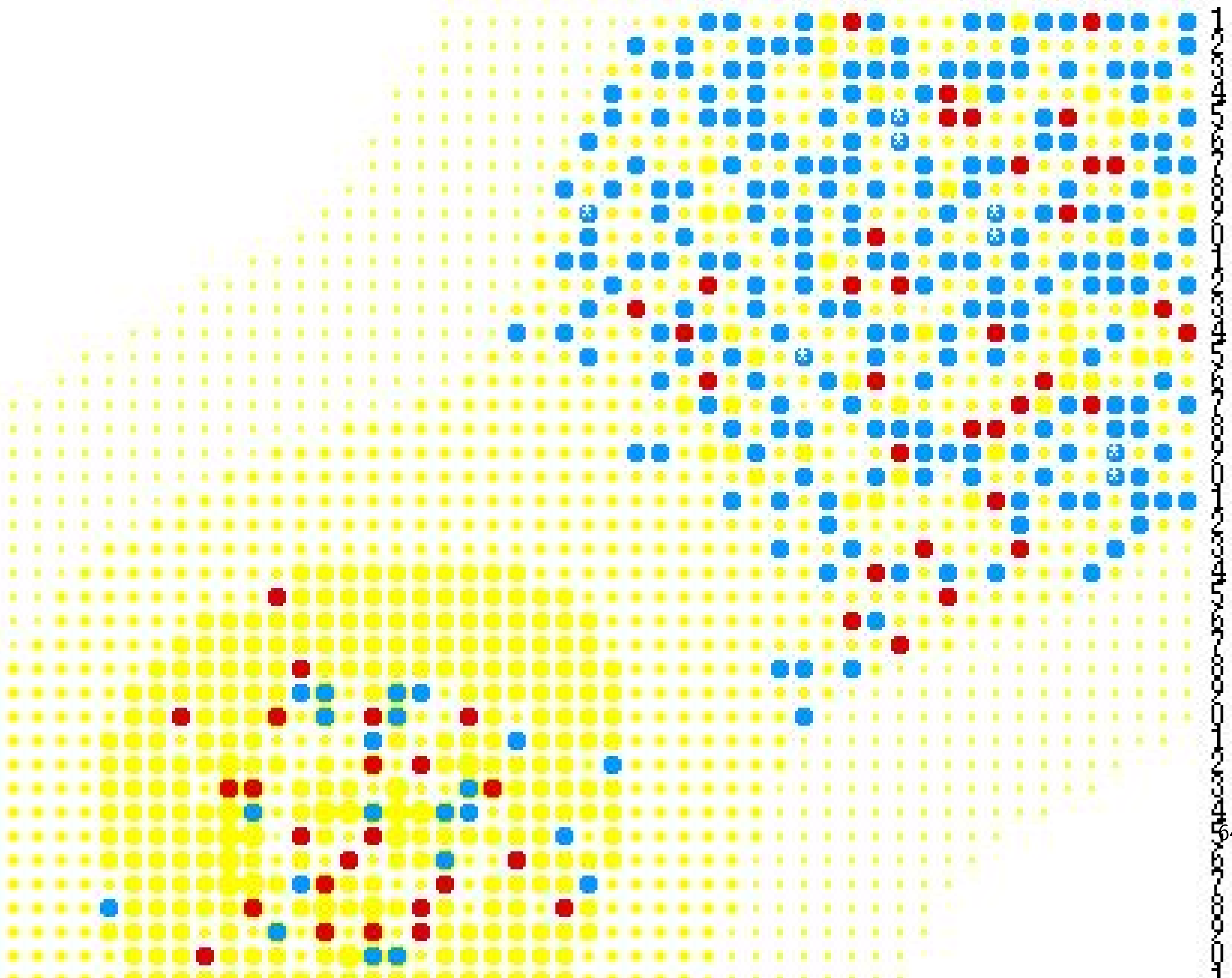
MCell. A very nice Java applet, with color and a library and randomization.

5. hensel.lifepatterns.net

   Very nice applet. Ranked first by Google, 3/4/03.

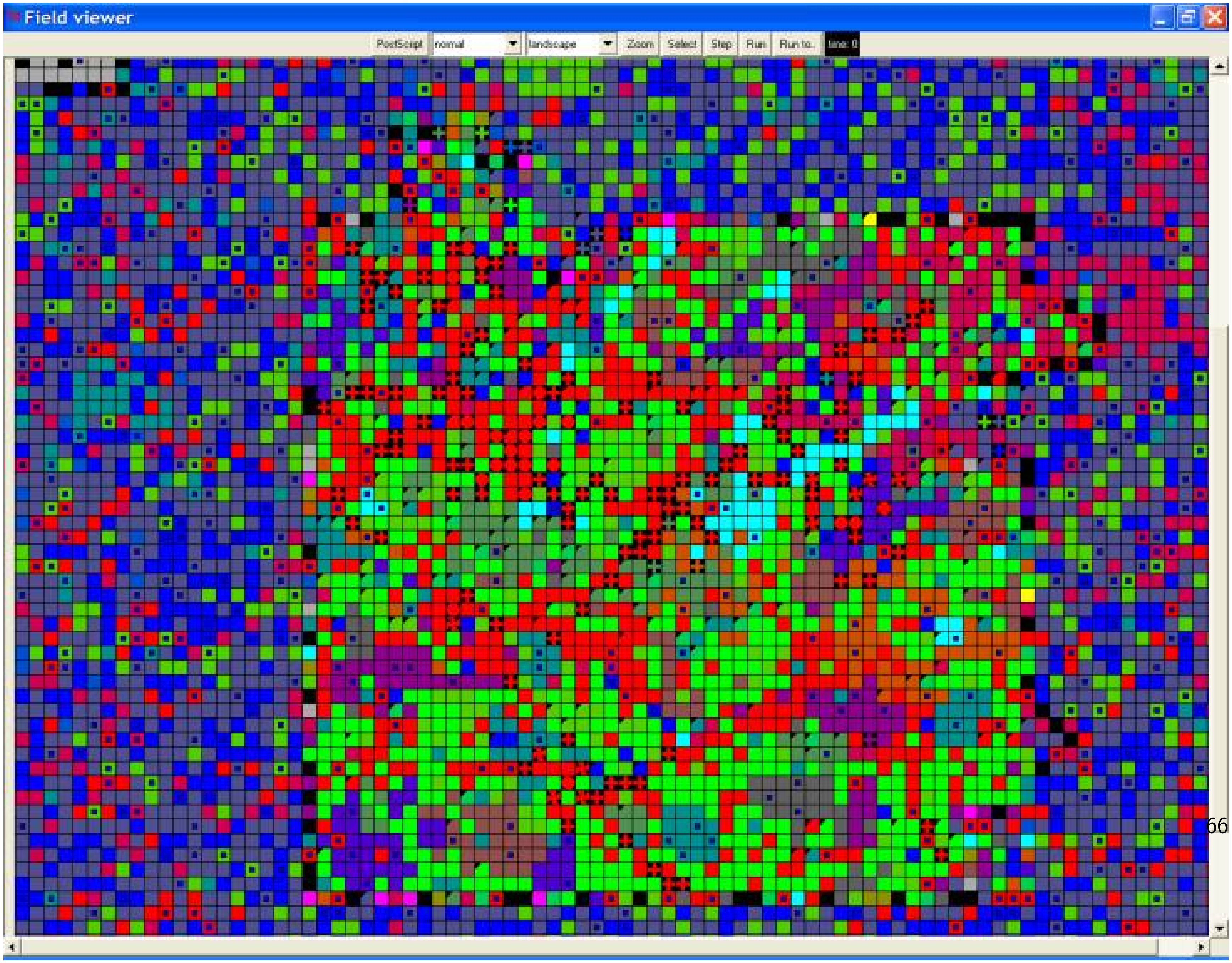
Sugarscape

- Epstein & Axtell, *Growing Artificial Societies*
- http://www.brook.edu/dybdocroot/SUGARSCAPE/



Agent-Based Simulation for Policy Studies

- Many examples; much activity
- Ian Lustick, PS-I
- http://www.ssc.upenn.edu/polisci/faculty/bios/lustic.html


Field viewer
PostScript
normal
landscape
Zoom
Select
Step
Run
Run to..
time: 0

Strategic Agents

- Games and strategic contexts
- Looking forward: fielding of strategic agents

  Beyond the bots of today. . .
- Limitations of game theory: finding the right strategy
- Limitations of game theory: failure to predict play by real players (human and not)

Prisoner's Dilemma

	C	D
C	3 3	5 0
D	0 5	1 1

Table 3: Prisoner's Dilemma in a Standard Form

Prisoner's Dilemma Parameterized: 1

	C	D
C	3, 3	0, $3+\delta$
D	$3+\delta$, 0	$3-\delta$, $3-\delta$

Table 4: Prisoner's Dilemma in a Cooperation-Adverse Form

Results from Two Reinforcement Learning Artificial Agents

δ	CC	CD	DC	DD	Cooperation by 1st player (out of 10,000)
0.05	253	122	129	9496	375
0.5	860	137	133	8870	997
1	300	124	112	9464	424
1.25	227	88	121	9564	315
1.5	1615	316	325	7744	1931
1.75	2900	1112	1085	4903	4012
2	2748	1681	1652	3919	4429
2.5	1919	2927	2988	2166	4846
2.95	905	4384	4199	512	5289

Prisoner’s Dilemma Parameterized: 2

	C	D
C	3 3	$3+\delta$ 0
D	0 $3+\delta$	δ δ

Table 5: Prisoner’s Dilemma in a Cooperation-Friendly Form

Results from Two Reinforcement Learning Artificial Agents

δ	CC	CD	DC	DD	Cooperation by 1st player (out of 10,000)
0.05	9453	267	258	22	9720
0.5	9230	312	314	144	9542
1	7591	674	633	1102	8265
1.25	4360	483	542	4615	4843
1.5	1297	456	443	7804	1753
1.75	2	112	100	9786	114
2	3	81	99	9817	84
2.5	4	87	101	9808	91
2.95	0	108	91	9801	108

In Conclusion. . .

Highlights of the Scene

- This is has been a *tour de horizon*—and a very selective one at that.
- AI is alive and flourishing in applications of e-commerce.
- Traditional topics/themes of encoding, structuring, and discovery remain valid.
- There has been, and continues to be, a flood of new and interesting ideas conforming to these themes.
- Application successes and opportunities abound.