How little I have learnt about search and why I care

Barry Richards

Edinburgh
HCRC / Cogitive Science Reunion
6th May 2010
(Edited for circulation)

Three questions (about AI)

- Why am I interested in search,

  particularly in intelligent search?

- Is the study of intelligent search,

  a **scientific** enterprise
  an **engineering** task
  an **indecipherable** quagmire?

- Have I learnt anything,

  that might be worth remembering?

Three traditional AI assumptions

- Search is the basic process / mechanism of AI.
- Heuristic search is the key to modelling intelligent search.
  - Intelligent search is necessary to solve NP-complete problems
- Intelligent search is one key to modelling intelligence.

Some challenges

- Evaluating a **search algorithm**
  - Three approaches
    1. Worst case complexity
    2. Average case performance
    3. Suitability / unsuitability for particular problem types / instances
  - Are there other approaches?

- Discriminating **intelligent search**
  - Newell and Simon's approach
    - Intelligent search is manifest in **Intelligent Problem Solving**
    - Specify the characteristics of Intelligent Problem Solving
  - Defining intelligent search
    - Is this possible?

- Finding appropriate **heuristic methods**
  - What differentiates a heuristic method as appropriate for intelligent search?

Four phases in my search

1. Blocks world planning
2. Large-scale commercial problems
3. Stochastic search
4. Constraint satisfaction problems

I. Blocks world planning

- Blocks world planning with **limited** resources
  - Resources
    - Table positions - Arms
  - Find a plan with
    - fewest actions / less than n actions
    - consuming least resources / no more than a fixed amount.

- *parc*PLAN
  - General planning architecture
  - **Hybrid** algorithm
    - Probe backing + CP (forward checking) + arc-consistency + heuristics

- Empirical tests
  - Random problem generator varying three parameters
    - Blocks - Table positions - Arms

What did I learn from the blocks world?

- *parc*PLAN
  - ✓ **Learnt** that it solved more test-bed problems faster than
    <u>some</u> other planners **<u>at</u>** **<u>that</u>** **<u>time</u>**.
  - ✓ **Learnt** how to combine some search techniques
    for the **<u>blocks</u>** **<u>world</u>**.
  - ✓ **Learnt** nothing that generalizes
    beyond the blocks world, except for one thing.

- **Summary**
  - ✓ Don’t solve most planning problems with *parc*PLAN.
    There is usually a much more efficient **special-purpose** algorithm.
  - ❑ This represents the **1st retreat** from generality.

II. Large-scale commercial problems

- Network optimization problems
  - Transport
    - Airlines (fleet planning, scheduling, tail assignment, revenue)
    - Logistics (distribution, resource allocation – construction, rescue & recovery)
    - Manpower planning
  - Telecommunications
    - Layer 3 routing (fast re-routing, BGP routing)
    - Bandwidth utilization,
    - Quality-of-Service,

What did I learn from commercial problems?

- From CHIP and ILOG
  - ✓ No “silver bullet”

- From ECLiPSe and ILOG Solver
  - ✓ “Bite the bullet”

- From ECLiPSe2 and OPL-CPLEX + ILOG Solver
  - ✓ Solve complex, multi-dimensional problems with **hybrid algorithms,**
    - Use a combination of SIMPLEX (for linear problems), finite domain techniques, stochastic algorithms, etc.

    NB. SIMPLEX (for some linear problems) has exponential worst case complexity, where it takes a number of steps exponential in the size of the problem.

What did I learn from commercial problems? (cont.)

- From mathematical programming (MP)
  - ✓ Don’t search in MP, **<u>sometimes</u>**.
    1. Integer programming (where all the variables are required to be integer) is typically NP-hard.
    2. Mixed integer programming (where only some of the variables are required to be integer) is also typically NP-hard.
    3. Performance profiles of integer and mixed integer programming reveal certain practical disadvantages that are severely limiting on some occasions.
    4. Finite domain reasoning, e.g. over the integers, is **<u>perhaps</u>** better done in constraint programming.
  - ✓ Don’t solve optimization problems in constraint programming, **<u>sometimes</u>**.

    e.g. those that can be linearized.
  - ✓ Use constraint programming for **<u>some</u>** scheduling problems,

    e.g. those that cannot be linearized.

What did I learn from commercial problems? (cont.)

- From constraint programming (CP)
  - ✓ Don't solve optimization problems in constraint programming, **sometimes**,
    e.g. those that can be linearized.
  - ✓ Use constraint programming for **some** scheduling problems,
    e.g. those that cannot be linearized. On search

    Some general "advice" on **some** techniques arising from **some** problems

- **Summary**
  - ✓ A programming "**cookbook**" for CP + MP
    - A **folklore**
      of how and when to apply **some** combinations of **some** techniques,
      derived from the experience of solving **some** problems
    - An **art**
      without predictable, or even probable, results.

❑ This represents the **2nd retreat** from generality.

III. Stochastic search

- Stochastic search paradigms
    - Simulated annealing
    - Evolutionary algorithms
    - Genetic algorithms

- Two “beliefs” about stochastic search
    - Well-suited to **optimization problems**,

      If there are lots of solutions to explore
    - Not so well suited to **decision problems,**

      If there are very few solutions.

Simulated annealing

- Basic algorithm structure
  a) Neighbourhood operator
     – Transforms one assignment into another
  b) Annealing schedule
     i. Initial 'acceptance probability' *AP (set very high)*
     ii. Schedule of increments for reducing *AP*
     iii. Terminating *AP*
     iv. Number of iterations of Neighbourhood-Operation at each increment.
  c) Objective function to be optimized

- Challenge for 'anytime' variant

  Find a **finite** annealing schedule that **balances** two factors.
     i. Gradient of reducing acceptance probability
     ii. Number of iterations of the neighbourhood operation at each increment.

  Search behaviour

  Fixing the **balance** determines the search profile.

Another “cookbook”

- On fixing the right balance
  - Right balance
    1. is **<u>specific</u>** to the problem (perhaps to the particular instance of the problem)
    2. Is assessed solely on the basis of empirical **<u>trials</u>**
    3. Is not derivable from a structural analysis of the problem (instance) since this is largely a **<u>black box</u>**.
- On altering search behaviour
  - a) Allow different annealing schedules for different parts of the search space
  - b) Bias the search towards certain <variable, value> pairs, depending upon the frequency of appearance in earlier iteration.

  NB Experiments show that both approaches can improve outcomes considerably.

What did I learn from stochastic algorithms?

- Understanding search in simulated annealing
  - ✓ The study is a **“dark art”** guided only by generate-and-test.
    - a) Improving search is a matter of art, beyond the reach of science.
    - b) Improving search results yields little insight into search behaviour

- **Summary**
  - ✓ Simulated annealing is typical of stochastic search.
  - ❑ This represents the **3rd retreat** from generality.

IV. Constraint satisfaction problems (CSPs)

- Three empirical **"observations"**
  1. Under-constrained instances
     - Often easy to solve,
       - because there are **lots** of solutions.
  2. Over-constrained instances
     - Sometimes easy to solve,
       - if the constraints prune (most) branches **early** in search.
     - Sometimes hard to solve,
       - if the constraints prune (most) branches only **late** in search.
  3. Critically-constrained instances
     - Typically very hard to solve,
       - because there are a **few** solutions but constraints prune (most) branches neither early nor late.

Conjectures and beliefs

- Three conjectures
  a) Critically-constrained problems typically coincide with a **peak** in search effort
  b) That **peak** typically coincides with a certain clause density (clause / variable ratio)
  c) That particular **clause density** can be identified statistically.

- Three beliefs
  d) Nonsystematic search algorithms are,
     1. poorly suited to large-scale problems with few solutions,
     2. either inefficient or inapplicable to unsolvable problems.

  e) Complete, backtrack-search techniques are effective for,
     1. tightly constrained problems with few solutions,
     2. over-constrained problems (no solutions).

  f) Incomplete, non-systematic techniques are effective for,
     large-scale problems with many solutions.

SAT problems and a local search algorithm

- 3-SAT problems
  - SAT problems in,
    - Conjunctive normal form
    - Each clause with exactly 3 variables
- ***learn-SAT*: A 3-SAT algorithm**
  - A complete, nonsystematic search
    - ✓ Learning no-goods
    - ✓ Learning-by-merging
  - Heuristics
    - ✓ Forward checking
    - ✓ Binary ordering
    - ✓ 3^{rd} order learning

3-SAT case study: Empirical tests and results

- Test problems – 3 types
  i. Problems with exactly <u>one</u> solution (AIM problems)
  ii. <u>Unsolvable</u> problems
  iii. Large-scale problems with <u>many</u> solutions

- Performance measure
  - Constraint density (clause / variable ratio) & number of constraint checks

- Test results – ***learn-SAT***
  i. **Outperforms** the best backtrack-search algorithms at lower clause densities,
  Otherwise "broadly" **approximates** them (e.g. *Tableau {CBJ + 3rd order learning}*)
  ii. **Outperforms** the best backtrack-search algorithms at lower clause densities
  Otherwise "broadly" **approximates** them (e.g. *relsat*)
  iii. **Approximates** the best local search algorithms at lower clause densities,
  Otherwise **grossly underperforms** them (e.g. *GSAT, weight*)

What did I learn from the case study?

- Three <u>beliefs</u> recalled

  d) Nonsystematic search algorithms are,

     1. poorly suited to large-scale problems with few solutions
     2. either inefficient or inapplicable to unsolvable problems.

  e) Complete, backtrack-search techniques are effective for,

     1. tightly constrained problems with few solutions,
     2. over-constrained problems (no solutions).

  f) Incomplete, non-systematic techniques are effective for,

     - large-scale problems with many solutions.

- *learn-SAT* results show,

  ✓ (d1) and (d2) are both **false**.

  ✓ (e1) and (e2) are **not true** at all clause densities, or even typically so.

  ✓ (f) is **relatively** true.

What did I learn from the case study? (cont.)

- Three <u>conjectures</u> recalled
  - a) Critically-constrained problems typically coincide with a **peak** in search effort
  - b) That **peak** typically coincides with a certain clause density (clause / variable ratio)
  - c) That particular **clause density** can be identified statistically.

- *learn-SAT* results show,
  - ✓ (a) is **false**.
    - Critically-constrained problems do **not** always coincide with a peak in search effort.
  - ✓ (b) is **false**.
    - Peak search effort does **not** always coincide with a particular clause density (clause / variable ratio).
  - ✓ (c) is **false**.
    - That reason for peak search **cannot** be identified statistically.

What did I learn from the case study? (cont.)

- On deriving insight into search
  - This depends essentially upon knowing **at least** 3 things.
    1. Some **parameters** of the problem

       e.g. constraint density, constraint width
    2. Something about the **solution space**

       e. g. number of solutions, distribution of solutions
    3. Some measure that **connects** (1) and (2) to performance,

       e.g. constraint checks

- **Summary**
  - On the prospects of understanding search
    - ✓ For most problems very little is known where (1) – (3) are satisfied.
  - ❑ This represents a **challenge**, not a retreat.

Why do I care about the challenge?

- AI problems

  Are mostly NP-Complete

  Require search to solve.

- Understanding search

  Remains a **“dark hole”** in AI research (as far as I can see).

- If AI is just engineering,

  I can live in the darkness, as long as the algorithms work to some specification.

- If AI is to be more than engineering,

  The darkness must be lifted.

Why do I care about the challenge? (cont.)

- AI cannot be a science,
  If the study of search is a "dark hole".
- The study of search can yield insight into intelligence,
  Only if it can be the subject of scientific enquiry.
- Scientific enquiry does not consist in,
  Folklore
  Programming cookbooks
  Anecdotes
- Questions
  i. Is **search** relevant to cognitive science?
  ii. Is **learning** a more relevant and promising horizon for cognitive science?