Dynamic Local Search for SAT: Design, Insights and Analysis

Dave Tompkins
Final PhD Oral Examination
Department of Computer Science, UBC
September 16, 2010

Supervisor: Holger Hoos
Supervisory Committee: Will Evans, Alan Hu
University Examiners: David Kirkpatrick, David Mitchell (SFU)
External Examiner: Steve Prestwich
Chair: Robin Turner (ECE)
Primary Goal

"to advance the state-of-the-art for SLS algorithms for SAT"
Primary Goal

"to advance the state-of-the-art for SLS algorithms for SAT"

• *Explicitly*: develop new SLS algorithms that can outperform existing algorithms
Primary Goal

"to advance the state-of-the-art for SLS algorithms for SAT"

- *Explicitly*: develop new SLS algorithms that can outperform existing algorithms

- *Implicitly*: advance our understanding of current algorithms and introduce tools for developing new algorithms
Overview

• Introduction
  – The Propositional Satisfiability problem (SAT)
  – Stochastic Local Search (SLS) for SAT
  – Summary of key contributions

• Body of Work

• Conclusions
  – Review key contributions
  – Future work
Propositional Satisfiability

• Boolean variables are either (T)rue or (F)alse
  – $x_1$: Dave's PhD defence will have a positive outcome
  – $x_2$: Dave will celebrate tonight

$\neg x_1 \lor x_2 \land (x_1 \lor \neg x_2)$

formul

negative literal

clause
Propositional Satisfiability

- Boolean variables are either (T)rue or (F)alse
  - $x_1$: Dave's PhD defence will have a positive outcome
  - $x_2$: Dave will celebrate tonight

<table>
<thead>
<tr>
<th>$x_1$</th>
<th>$x_2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>F</td>
<td>T</td>
</tr>
<tr>
<td>T</td>
<td>F</td>
</tr>
<tr>
<td>T</td>
<td>T</td>
</tr>
</tbody>
</table>

$(\neg x_1 \lor x_2) \land (x_1 \lor \neg x_2)$
Propositional Satisfiability

- Boolean variables are either (T)rue or (F)alse
  - $x_1$: Dave's PhD defence will have a positive outcome
  - $x_2$: Dave will celebrate tonight

<table>
<thead>
<tr>
<th>$x_1$</th>
<th>$x_2$</th>
<th>$(\neg x_1 \lor x_2) \land (x_1 \lor \neg x_2)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>F</td>
<td>F</td>
<td>$T$</td>
</tr>
<tr>
<td>F</td>
<td>T</td>
<td>$F$</td>
</tr>
<tr>
<td>T</td>
<td>F</td>
<td>$F$</td>
</tr>
<tr>
<td>T</td>
<td>T</td>
<td>$T$</td>
</tr>
</tbody>
</table>
Propositional Satisfiability

• Boolean variables are either (T)rue or (F)alse
  – $x_1$: Dave's PhD defence will have a positive outcome
  – $x_2$: Dave will celebrate tonight

<table>
<thead>
<tr>
<th>$x_1$</th>
<th>$x_2$</th>
<th>$(\neg x_1 \lor x_2) \land (x_1 \lor \neg x_2)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>F</td>
<td>F</td>
<td>$T$</td>
</tr>
<tr>
<td>F</td>
<td>T</td>
<td>$F$</td>
</tr>
<tr>
<td>T</td>
<td>F</td>
<td>$F$</td>
</tr>
<tr>
<td>T</td>
<td>T</td>
<td>$T$</td>
</tr>
</tbody>
</table>

• Objective: Given a formula (SAT instance) find a satisfying assignment
Many "Real" SAT Applications

Software Verification

Sudoku

8 4 6
1
5 9 3
7
4 8 2
1
3 9 2 5
4 6 5
7 8
7
1 3
9
Exponential Search Space

<table>
<thead>
<tr>
<th>$x_1$</th>
<th>$x_2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>F</td>
<td>T</td>
</tr>
<tr>
<td>T</td>
<td>F</td>
</tr>
<tr>
<td>T</td>
<td>T</td>
</tr>
</tbody>
</table>
Exponential Search Space

<table>
<thead>
<tr>
<th>$x_1$</th>
<th>$x_2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>F</td>
<td>T</td>
</tr>
<tr>
<td>T</td>
<td>F</td>
</tr>
<tr>
<td>T</td>
<td>T</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$x_1$</th>
<th>$x_2$</th>
<th>$x_3$</th>
</tr>
</thead>
<tbody>
<tr>
<td>F</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>F</td>
<td>F</td>
<td>T</td>
</tr>
<tr>
<td>F</td>
<td>T</td>
<td>F</td>
</tr>
<tr>
<td>T</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>T</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>T</td>
<td>F</td>
<td>T</td>
</tr>
<tr>
<td>T</td>
<td>T</td>
<td>F</td>
</tr>
<tr>
<td>T</td>
<td>T</td>
<td>T</td>
</tr>
</tbody>
</table>
Exponential Search Space
Exponential Search Space

• $n$ variables:
  $2^n$ assignments

• 250 variables
  $\approx 10^{75}$ combinations
  $\approx \# \text{ atoms in the universe}$
Stochastic Local Search (SLS) for SAT

randomly initialize all variables
while (formula not satisfied)
    select a variable and “flip” it
**Stochastic Local Search (SLS) for SAT**

randomly initialize all variables
while (formula not satisfied)
select a variable and “flip” it

\[
\begin{array}{c|c|c|c|c|c}
  x_1 & x_2 & x_3 & x_4 & x_5 \\
  \hline
  \neg x_1 \lor x_2 \lor \neg x_5 \land \neg x_1 \lor \neg x_2 \lor x_4 \land \neg x_4 \lor \neg x_5 \land \neg x_1 \lor x_2 \lor x_3 \lor \neg x_4 \\
\end{array}
\]
randomly initialize all variables
while (formula not satisfied)
  select a variable and “flip” it

<table>
<thead>
<tr>
<th>$x_1$</th>
<th>$x_2$</th>
<th>$x_3$</th>
<th>$x_4$</th>
<th>$x_5$</th>
</tr>
</thead>
<tbody>
<tr>
<td>T</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>T</td>
</tr>
</tbody>
</table>

$$(\neg x_1 \lor x_2 \lor \neg x_5) \land (\neg x_1 \lor \neg x_2 \lor x_4) \land (\neg x_4 \lor \neg x_5) \land (\neg x_1 \lor x_2 \lor x_3 \lor \neg x_4)$$

$$(\neg x_1 \lor x_2 \lor \neg x_5) \land (\neg x_1 \lor \neg x_2 \lor x_4) \land (\neg x_4 \lor \neg x_5) \land (\neg x_1 \lor x_2 \lor x_3 \lor \neg x_4)$$
Stochastic Local Search (SLS) for SAT

randomly initialize all variables
while (formula not satisfied)
select a variable and “flip” it

<table>
<thead>
<tr>
<th>$x_1$</th>
<th>$x_2$</th>
<th>$x_3$</th>
<th>$x_4$</th>
<th>$x_5$</th>
</tr>
</thead>
<tbody>
<tr>
<td>T</td>
<td>F</td>
<td>F</td>
<td>T</td>
<td>T</td>
</tr>
</tbody>
</table>

$(\neg x_1 \lor x_2 \lor \neg x_3) \land (\neg x_1 \lor \neg x_2 \lor x_4) \land (\neg x_4 \lor \neg x_5) \land (\neg x_1 \lor x_2 \lor x_3 \lor \neg x_4)$

$(\neg x_1 \lor x_2 \lor \neg x_5) \land (\neg x_1 \lor \neg x_2 \lor x_4) \land (\neg x_4 \lor \neg x_5) \land (\neg x_1 \lor x_2 \lor x_3 \lor \neg x_4)$
Stochastic Local Search (SLS) for SAT

randomly initialize all variables
while (formula not satisfied)
select a variable and “flip” it

<table>
<thead>
<tr>
<th>$x_1$</th>
<th>$x_2$</th>
<th>$x_3$</th>
<th>$x_4$</th>
<th>$x_5$</th>
</tr>
</thead>
<tbody>
<tr>
<td>T</td>
<td>F</td>
<td>F</td>
<td>T</td>
<td>T</td>
</tr>
</tbody>
</table>

$(\neg x_1 \lor x_2 \lor \neg x_5) \land (\neg x_1 \lor \neg x_2 \lor x_4) \land (\neg x_4 \lor \neg x_5) \land (\neg x_1 \lor x_2 \lor x_3 \lor \neg x_4)$

$(\neg x_1 \lor x_2 \lor \neg x_5) \land (\neg x_1 \lor \neg x_2 \lor x_4) \land (\neg x_4 \lor \neg x_5) \land (\neg x_1 \lor x_2 \lor x_3 \lor \neg x_4)$

$(\neg x_1 \lor x_2 \lor \neg x_5) \land (\neg x_1 \lor \neg x_2 \lor x_4) \land (\neg x_4 \lor \neg x_5) \land (\neg x_1 \lor x_2 \lor x_3 \lor \neg x_4)$
Stochastic Local Search (SLS) for SAT

randomly initialize all variables
while (formula not satisfied)
select a variable and “flip” it

<table>
<thead>
<tr>
<th>$x_1$</th>
<th>$x_2$</th>
<th>$x_3$</th>
<th>$x_4$</th>
<th>$x_5$</th>
</tr>
</thead>
<tbody>
<tr>
<td>T</td>
<td>F</td>
<td>F</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>T</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>T</td>
</tr>
<tr>
<td>T</td>
<td>T</td>
<td>F</td>
<td>F</td>
<td>T</td>
</tr>
</tbody>
</table>

$(¬x_1 ∨ x_2 ∨ ¬x_5) ∧ (¬x_1 ∨ ¬x_2 ∨ x_4) ∧ (¬x_4 ∨ ¬x_5) ∧ (¬x_1 ∨ x_2 ∨ x_3 ∨ ¬x_4)$
Stochastic Local Search (SLS) for SAT

randomly initialize all variables
while (formula not satisfied)
select a variable and “flip” it

<table>
<thead>
<tr>
<th>(x_1)</th>
<th>(x_2)</th>
<th>(x_3)</th>
<th>(x_4)</th>
<th>(x_5)</th>
</tr>
</thead>
<tbody>
<tr>
<td>T</td>
<td>F</td>
<td>F</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>T</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>T</td>
<td>T</td>
<td>F</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>F</td>
<td>T</td>
<td>F</td>
<td>F</td>
<td>T</td>
</tr>
</tbody>
</table>

\[ (\neg x_1 \lor x_2 \lor \neg x_5) \land (\neg x_1 \lor \neg x_2 \lor x_4) \land (\neg x_4 \lor \neg x_5) \land (\neg x_1 \lor \neg x_2 \lor \neg x_3 \lor \neg x_4) \]

\[ (\neg x_1 \lor x_2 \lor \neg x_5) \land (\neg x_1 \lor \neg x_2 \lor x_4) \land (\neg x_4 \lor \neg x_5) \land (\neg x_1 \lor x_2 \lor x_3 \lor \neg x_4) \]

\[ (\neg x_1 \lor x_2 \lor \neg x_5) \land (\neg x_1 \lor \neg x_2 \lor x_4) \land (\neg x_4 \lor \neg x_5) \land (\neg x_1 \lor x_2 \lor x_3 \lor \neg x_4) \]

\[ (\neg x_1 \lor x_2 \lor \neg x_5) \land (\neg x_1 \lor \neg x_2 \lor x_4) \land (\neg x_4 \lor \neg x_5) \land (\neg x_1 \lor x_2 \lor x_3 \lor \neg x_4) \]
Stochastic Local Search (SLS) for SAT

randomly initialize all variables
while (formula not satisfied)
    select a variable and “flip” it

• Selecting a variable:
  make = # of clauses that become satisfied if we flip \( x \)
  break = ... 
  score = make − break [GSAT: Selman, Levesque & Mitchell, 1992]
Key Contributions
Key Contributions

1. Developed UBCSAT
2. Created SAPS, a Clause Penalty (CP) algorithm
3. Analyzed CP algorithm behaviour
4. Analyzed random decisions in SLS algorithms
5. Introduced a new conceptual model for SLS algorithms with Variable Expressions (VEs)
   – Developed a new Design Architecture (DAVE)
Key Contributions

1. Developed UBCSAT
2. Created SAPS, a Clause Penalty (CP) algorithm
3. Analyzed CP algorithm behaviour
4. Analyzed random decisions in SLS algorithms
5. Introduced a new conceptual model for SLS algorithms with Variable Expressions (VEs)
   – Developed a new Design Architecture (DAVE)
UBCSAT Architecture

while (formula not satisfied)
    select a variable and "flip" it
UBCSAT Architecture

while (formula not satisfied)
select a variable and "flip" it
UBCSAT Algorithms

- All (typical) SLS algorithms can be seen as a series of procedures that happen at "event points"

- When you select the algorithm, the appropriate procedures are "triggered"
UBCSAT Algorithms

- Similar algorithms can re-use existing triggers
• Additional Reports and Statistics can be "activated" when needed
UBCSAT Reports

- Facilitating empirical analysis is an important component of UBCSAT
UBCSAT Efficiency

- UBCSAT is very efficient with little overhead

<table>
<thead>
<tr>
<th>algorithm</th>
<th>UBCSAT Speedup</th>
</tr>
</thead>
<tbody>
<tr>
<td>WalkSAT/SKC</td>
<td>1.5x – 2.2x</td>
</tr>
<tr>
<td>Novelty</td>
<td>1.3x – 2.0x</td>
</tr>
<tr>
<td>GSAT</td>
<td>1.7x – 7.6x</td>
</tr>
<tr>
<td>GWSAT</td>
<td>2.5x – 7.4x</td>
</tr>
</tbody>
</table>
UBCSAT

A software framework for SLS algorithms

• Incorporates existing SLS algorithms
  – highly efficient, accurate implementations
• Facilitate development of new SLS algorithms
• Advanced empirical analysis of algorithms
• Open-source
• Cornerstone of the dissertation
Key Contributions

1. Developed UBCSAT
2. Created SAPS, a Clause Penalty (CP) algorithm
3. Analyzed CP algorithm behaviour
4. Analyzed random decisions in SLS algorithms
5. Introduced a new conceptual model for SLS algorithms with Variable Expressions (VEs)
   - Developed a new Design Architecture (DAVE)
SAT Search Space

• n-dimensional hypercube
"2D" Search Landscape

# unsat clauses

solution
"2D" Search Landscape

- # unsat clauses
- Local minimum
- Solution
Intensification & Diversification
Clause Penalties

- Each clause is assigned a penalty value

- Score is no longer just make – break
  score = $\sum_{\text{make}} \text{c.penalty} - \sum_{\text{break}} \text{c.penalty}$

Original Idea:
- Breakout Method [Morris, 1993]
- GSAT+CW [Selman & Kautz, 1993]
"Breakout" Approach

- When a local minimum occurs:

\[ \sum_{\text{make}} \text{c.penalty} \leq \sum_{\text{break}} \text{c.penalty} \]

increment the penalty for unsatisfied clauses
"Breakout" Approach

• When a local minimum occurs:
  \[ \sum_{\text{make}} \text{c.penalty} \leq \sum_{\text{break}} \text{c.penalty} \]
  increment the penalty for unsatisfied clauses

• Eventually, will no longer be in a local minimum

  \[ \text{score} = \sum_{\text{make}} \text{c.penalty} - \sum_{\text{break}} \text{c.penalty} \]
"Breakout" Approach

• When a local minimum occurs:
  \[ \sum_{\text{make}} \text{c.penalty} \leq \sum_{\text{break}} \text{c.penalty} \]
  increment the penalty for unsatisfied clauses

• Eventually, will no longer be in a local minimum
  \[
  \text{score} = \sum_{\text{make}} \text{c.penalty} - \sum_{\text{break}} \text{c.penalty}
  \]
SAPS Algorithm

• Enhancement of existing algorithm
  – Exponentiated Sub-Gradient (ESG)
    [Schuurmans et. al, 2002]

• Multiplicative Scaling
  \[
  c\text{.penalty} := c\text{.penalty} \cdot \alpha
  \]

• Probabilistic Smoothing
  with probability (Ps):
  \[
  c\text{.penalty} := c\text{.penalty} + (1-\rho) \cdot \text{avg.penalty}
  \]

• Scaling And Probabilistic Smoothing (SAPS)
SAPS Algorithm

• Dominated the performance of its predecessor (ESG)

• Still amongst the state-of-the-art solvers

• Led to the work in other chapters
1. Developed UBCSAT
2. Created SAPS, a Clause Penalty (CP) algorithm
3. Analyzed CP algorithm behaviour
4. Analyzed random decisions in SLS algorithms
5. Introduced a new conceptual model for SLS algorithms with Variable Expressions (VEs)
   - Developed a new Design Architecture (DAVE)
Dynamic Clause Penalties
Clause Penalty Distributions
Clause Penalty Analysis

• We identified instances with "Problem Clauses"
  – We constructed *weighted* instances...
    ... that were *easier* for SLS algorithms to solve
      (80x faster for Adaptive Novelty+)
Clause Penalty Analysis

- A quest for problem clauses
- Analyzed penalty behaviour
- Hardness of warped landscapes
- History ("memory") of the search

- Ultimately: problem clauses are rarely helpful
- Key element of CP algorithms: diversification
Key Contributions

1. Developed UBCSAT
2. Created SAPS, a Clause Penalty (CP) algorithm
3. Analyzed CP algorithm behaviour
4. Analyzed random decisions in SLS algorithms
5. Introduced a new conceptual model for SLS algorithms with Variable Expressions (VEs)
   – Developed a new Design Architecture (DAVE)
Random Decisions

- **Stochastic Local Search**
- **Quality** of random decision
  - SLS Algorithms are robust (existing random number generators are good enough)
- **Quantity** of random decisions
  - Simple derandomizations can be effective
  - SLS Algorithms exhibit 'chaotic'-like behaviour
  - No real advantage to derandomizing
Key Contributions

1. Developed UBCSAT
2. Created SAPS, a Clause Penalty (CP) algorithm
3. Analyzed CP algorithm behaviour
4. Analyzed random decisions in SLS algorithms
5. Introduced a new conceptual model for SLS algorithms with Variable Expressions (VEs)
   - Developed a new Design Architecture (DAVE)
Variable Properties

- **Scoring Properties**
  - \( \text{make} = \) \# of clauses that become satisfied if we flip \( x \)
  - \( \text{break} = \) \# of clauses that become unsatisfied if we flip \( x \)
  - \( \text{score} = \text{make} - \text{break} \)
Variable Properties

• **Scoring Properties**
  make = # of clauses that become satisfied if we flip $x$
  break = ... unsatisfied ... 
  score = make – break

• **Dynamic Properties**
  age = # of steps since $x$ was flipped [TABU, Glover 1986]
  flips = # of times $x$ has been flipped [HSAT, Gent & Walsh 1992]
Variable Properties

• Scoring Properties
  make = # of clauses that become satisfied if we flip $x$
  break = ... unsatisfied ... 
  score = make – break

• Dynamic Properties
  age = # of steps since $x$ was flipped [TABU, Glover 1986]
  flips = # of times $x$ has been flipped [HSAT, Gent & Walsh 1992]

• Static Properties
Variable Expressions (VEs)

- combinations of variable properties in mathematical expressions:

  make – break
  age
  (make – break) + 3 \cdot \log_2(\text{age}) + \text{age}/\text{flips}

- Most existing SLS algorithms use straightforward VEs
  ... we explore more complex VEs

- Our work was inspired by:
  Variable Weighting Algorithm VW2 [Prestwich, 2005]
Combining Properties

Select variable with minimum value of:

\[ \text{break} + c \cdot \text{flips} \]
Combining Properties

Select variable with minimum value of:

\[ \text{break} + c \cdot \text{flips} \]
Combining Properties

• **Normalize** properties values to [0...1] amongst the “candidate” variables

• Allow for **non-linear** normalization
Modifying Existing Algorithms with VEs

- WalkSAT with more complex VE

- Speedup factor:
  7.2x (steps)
  3.1x (time)

- (compared to original WalkSAT)
  > 4000x (steps)
  > 2000x (time)
Our New SLS Model

Filter Variables → Variable Expression(s) → Selection Mechanism
Our New SLS Model

Filter Variables → Variable Expression(s) → Selection Mechanism
Separation of:
VEs & Selection Mechanism

• Novelty Algorithm [McAllester, Selman & Kautz, 1997]

• Select “best” variable with maximum of:
  (make – break)
  breaking ties by
  (age)

• If the best variable has the minimum
  (age)
  then, with probability p, select 2^{nd} best var.
Separation of:
**VEs & Selection Mechanism**

- **Novelty Algorithm** [McAllester, Selman & Kautz, 1997]

- Select “best” variable with maximum of:
  \[(\text{VE}_1)\]
  breaking ties by
  \[(\text{VE}_2)\]

- If the best variable has the minimum
  \[(\text{VE}_3)\]
  then, with probability \(p\), select 2\(^{nd}\) best var.
Our New SLS Model

Filter Variables → Evaluate VEs → Select Variable
Our New SLS Model

Algorithm Controller: Determine the Filter, VEs & VSM

Filter Variables → Evaluate VEs → Select Variable

Flip Selected Variable & Update State Information / Bookkeeping
Our New SLS Model

Algorithm Controller: Determine the Filter, VEs & VSM

- Filter Variables
- Evaluate VEs
- Select Variable
- Flip Selected Variable & Update State Information / Bookkeeping
Algorithm Controllers

Controller
  - FILT
  - FILT
  - FILT
    - VEs
    - VEs
    - VEs
    - VSM
    - VSM
    - VSM
Algorithm Controllers

Controller

FILT → VEs → VSM
FILT → VEs → VSM
FILT → VEs → VSM

Controller

FILT
VEs
VSM

Sub-Controller
Sub-Controller
Sub-Controller
Our New SLS Model

Algorithm Controller: Determine the Filter, VEs & VSM

- Filter Variables
- Evaluate VEs
- Select Variable

Flip Selected Variable & Update State Information / Bookkeeping
Software Implementation

• Design Architecture for Variable Expressions (DAVE)
  – Entire algorithm specified at runtime
    • Controllers, filters, VEs, selection mechanisms
  – Arbitrary complex VEs (interpreted)
  – Sophisticated macro system
    • Aids the use of automated configurators

• Extension of UBCSAT (2.0)
New Model & DAVE

• Concept of VEs
• New Model
• New Architecture

• Demonstrated our work in conjunction with an automated configurator

• Speedup factor:
  16.2x (steps)
  9.0x (time)
Key Contributions

1. Developed UBCSAT
2. Created SAPS, a Clause Penalty (CP) algorithm
3. Analyzed CP algorithm behaviour
4. Analyzed random decisions in SLS algorithms
5. Introduced a new conceptual model for SLS algorithms with Variable Expressions (VEs)
   - Developed a new Design Architecture (DAVE)
Primary Goal

"to advance the state-of-the-art for SLS algorithms for SAT"

- *Explicitly*: develop new SLS algorithms that can outperform existing algorithms

- *Implicitly*: advance our understanding of current algorithms and introduce tools for developing new algorithms
Future Work

- Extend our methods to other domains
- Incorporate the use of automated tools
- Dynamic instances, distributed systems
- Generalized clause penalty solver
- Problem clauses & encodings
- New algorithm constructions
Selected Publications


Special Thanks To:

- Supervisor: Holger H. Hoos
Special Thanks To:

• Committee members:
  – Will Evans, Alan Hu (& Lee Iverson)
• Co-Authors:
  – Holger H. Hoos & Frank Hutter
• Additional technical help
  – Kevin Smyth, Lin Xu, Chris Fawcett
• BETA lab members
• Proofreaders
• Family & friends
Special Thanks To:
Questions...