Finding the Right Words

Everything you always wanted to know about Optimality Theory, Harmony Grammar and Simulated Annealing, but were afraid to ask

Tamás Biró

ACLC, University of Amsterdam (UvA)

Sound Circle, May 14, 2009

Everything you always wanted to know about OT, HG and SA

- OT: Optimality Theory
- HG: Harmony Grammar
- SA: Simulated Annealing – an implementation

Warning:

- Not much new for computational linguists.
- Not much new for those familiar with my past work.
- I'm misleading you.
  Nothing connectionist, only symbolic approaches.

Everything you always wanted to know about OT, HG and SA

- OT: Optimality Theory
- HG: Harmony Grammar
- SA: Simulated Annealing – an implementation

Warning:

- Not much new for computational linguists.
- Not much new for those familiar with my past work.
- I'm misleading you.
  Nothing connectionist, only symbolic approaches.

Overview

1. OT and HG
2. Implementing HG and OT
3. SA-OT: Simulated Annealing for Optimality Theory
4. An example
5. Conclusions

Overview

1. OT and HG
2. Implementing HG and OT
3. SA-OT: Simulated Annealing for Optimality Theory
4. An example
5. Conclusions

The idea of optimizing

Optimizing disciplines since the 18th century:

- Physics: minimize energy, maximize entropy, etc.
- Economics: minimize costs.
- Evolutionary biology: maximize fitness.

Simple model to work with mathematically. Lots of past work:

- Elementary calculus, Lagrange multipliers, linear programming, etc.
- Exact and heuristic optimization algorithms.

Basics of optimization

Two ingredients of finding the highest point in a landscape:

- *Search space*: set W ("horizontal structure").
- *Target function*: $H(w)$, where $w \in W$ ("vertical structure").

The "solution" is the $w^* \in W$ such that $H(w^*)$ is "the best":

$$w^* = \arg\operatorname*{opt}_{w \in W} H(w)$$

For instance, if maximizing $H(w)$, then $H(w^*) \geq H(w)$ for all $w \in W$.

Optimization in linguistics

Generative linguistics: how to map U onto $\mathrm{SF}(U)$?

$$
\begin{aligned}
w^* &= \arg\operatorname*{opt}_{w \in W} H(w) \\
\mathrm{SF}(U) &= \arg\operatorname*{opt}_{w \in \mathrm{Gen}(U)} H(w)
\end{aligned}
$$

Ingredients of optimization in linguistics:

- *Search space*: possible forms (candidates).
- *Target function*: “Harmony”.

The Harmony function

What is “Harmony”?

Whatever is $H(w)$, its range must be ranked:

$H(w_1) \geq H(w_2)$ or $H(w_2) \geq H(w_1)$. How to do that?

- Elementary functions (related to linguistic features) to be optimized: $C_i(w)$ (aka “constraints”, a misnomer for historical reasons).
- For instance, $C_i(w) \in \mathbb{N}_0$. (Not always.)
- How to build $H(w)$ from several $C_i(w)$'s?

Building the Harmony function

1. Summing up: $H(w) = C_1(w) + C_2(w) + ... + C_N(w)$
2. Weighted sum: $H(w) = g_1 \cdot C_1(w) + g_2 \cdot C_2(w) + ... + g_N \cdot C_N(w)$
3. OT tableau row: $H(w) =$ | $C_1(w)$ | $C_2(w)$ | ... | $C_N(w)$ |
4. Exponential weights: $-H(w) = C_1(w) \cdot q^N + C_2(w) \cdot q^{N-1} + ... + C_N(w) \cdot q$
5. (Hard constraints: $H(w) = C_1(w)$ & $C_2(w)$ & ... & $C_N(w)$)

Comparing $H(w_1)$ to $H(w_2)$:

- HG: 1,2,4: comparing real values.
- OT: 3: use lexicographic order (NB: cf. 4. $q \rightarrow +\infty$).

NB: two interpretations of "unordered constraints" in OT: same weight vs. both permutations.

Building the Harmony function

1. Summing up: $H(w) = C_1(w) + C_2(w) + ... + C_N(w)$
2. Weighted sum: $H(w) = g_1 \cdot C_1(w) + g_2 \cdot C_2(w) + ... + g_N \cdot C_N(w)$
3. OT tableau row: $H(w) =$ | $C_1(w)$ | $C_2(w)$ | ... | $C_N(w)$ |
4. Exponential weights: $-H(w) = C_1(w) \cdot q^N + C_2(w) \cdot q^{N-1} + ... + C_N(w) \cdot q$
5. (Hard constraints: $H(w) = C_1(w)$ & $C_2(w)$ & ... & $C_N(w)$)

Comparing $H(w_1)$ to $H(w_2)$:

- HG: 1,2,4: comparing real values.
- OT: 3: use lexicographic order (NB: cf. 4. $q \rightarrow +\infty$).

NB: two interpretations of "unordered constraints" in OT: same weight vs. both permutations.

Building the Harmony function

1. Summing up: $H(w) = C_1(w) + C_2(w) + ... + C_N(w)$
2. Weighted sum: $H(w) = g_1 \cdot C_1(w) + g_2 \cdot C_2(w) + ... + g_N \cdot C_N(w)$
3. OT tableau row: $H(w) =$

$C_1(w)$	$C_2(w)$	...	$C_N(w)$

4. Exponential weights: $-H(w) = C_1(w) \cdot q^N + C_2(w) \cdot q^{N-1} + ... + C_N(w) \cdot q$
5. (Hard constraints: $H(w) = C_1(w)$ & $C_2(w)$ & ... & $C_N(w)$)

Comparing $H(w_1)$ to $H(w_2)$:

- HG: 1,2,4: comparing real values.
- OT: 3: use lexicographic order (NB: cf. 4. $q \to +\infty$).

NB: two interpretations of "unordered constraints" in OT: same weight vs. both permutations.

Building the Harmony function

1. Summing up: $H(w) = C_1(w) + C_2(w) + ... + C_N(w)$
2. Weighted sum: $H(w) = g_1 \cdot C_1(w) + g_2 \cdot C_2(w) + ... + g_N \cdot C_N(w)$
3. OT tableau row: $H(w) =$

$C_1(w)$	$C_2(w)$	...	$C_N(w)$

4. Exponential weights: $-H(w) = C_1(w) \cdot q^N + C_2(w) \cdot q^{N-1} + ... + C_N(w) \cdot q$
5. (Hard constraints: $H(w) = C_1(w)$ & $C_2(w)$ & ... & $C_N(w)$)

Comparing $H(w_1)$ to $H(w_2)$:

- HG: 1,2,4: comparing real values.
- OT: 3: use lexicographic order (NB: cf. 4. $q \to +\infty$).

NB: two interpretations of “unordered constraints” in OT: same weight vs. both permutations.

Building the Harmony function

1. Summing up: $H(w) = C_1(w) + C_2(w) + ... + C_N(w)$
2. Weighted sum: $H(w) = g_1 \cdot C_1(w) + g_2 \cdot C_2(w) + ... + g_N \cdot C_N(w)$
3. OT tableau row: $H(w) =$ | $C_1(w)$ | $C_2(w)$ | ... | $C_N(w)$ |
4. Exponential weights: $-H(w) = C_1(w) \cdot q^N + C_2(w) \cdot q^{N-1} + ... + C_N(w) \cdot q$
5. (Hard constraints: $H(w) = C_1(w)$ & $C_2(w)$ & ... & $C_N(w)$)

Comparing $H(w_1)$ to $H(w_2)$:

- HG: 1,2,4: comparing real values.
- OT: 3: use lexicographic order (NB: cf. 4. $q \rightarrow +\infty$).

NB: two interpretations of "unordered constraints" in OT: same weight vs. both permutations.

Building the Harmony function

1. Summing up: $H(w) = C_1(w) + C_2(w) + \ldots + C_N(w)$
2. Weighted sum: $H(w) = g_1 \cdot C_1(w) + g_2 \cdot C_2(w) + \ldots + g_N \cdot C_N(w)$
3. OT tableau row: $H(w) =$ $\boxed{C_1(w) \mid C_2(w) \mid \ldots \mid C_N(w)}$
4. Exponential weights: $-H(w) = C_1(w) \cdot q^N + C_2(w) \cdot q^{N-1} + \ldots + C_N(w) \cdot q$
5. (Hard constraints: $H(w) = C_1(w)$ & $C_2(w)$ & ... & $C_N(w)$)

Comparing $H(w_1)$ to $H(w_2)$:

- HG: 1,2,4: comparing real values.
- OT: 3: use lexicographic order (NB: cf. 4. $q \rightarrow +\infty$).

NB: two interpretations of “unordered constraints” in OT: same weight vs. both permutations.

Overview

1. OT and HG
2. Implementing HG and OT
3. SA-OT: Simulated Annealing for Optimality Theory
4. An example
5. Conclusions

A function and its implementation

- Generative linguistics: grammar is a **map** from U to $\mathrm{SF}(U)$.
- Linguistic competence $=$ grammar.
- **Implementation:** an algorithm that finds for each U the corresponding

$$\mathrm{SF}(U) = \underset{w \in \mathrm{Gen}(U)}{\arg\,\mathrm{opt}}\ H(w)$$

  (for given Gen and H).
- Use in language technology, etc.
- Linguistic performance $=$ implementation of grammar.
- Modelling linguistic performance: e.g., speech errors in fast speech.

Existing implementations of Optimality Theory

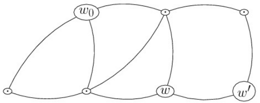
How can the optimal candidate be found?

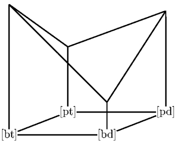
- finite-state OT (Ellison, Eisner, Karttunen, Frank & Satta, Gerdemann & van Noord, Jäger...)
- chart parsing (dynamic programing) (Tesar & Smolensky; Kuhn)

These are perfect for language technology: they always find the optimal candidate (if conditions met!).
But we would like a psychologically adequate model of linguistic performance including performance errors: **Simulated Annealing.**

Overview

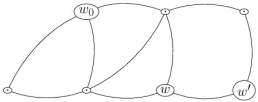
Basic idea of Simulated Annealing

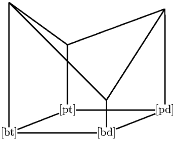




- Neighbourhood structure on the candidate set.
- Random walk. If neighbour more optimal: move.
- If less optimal: move in the beginning, don't move later.
- Neighbourhood structure $\rightarrow$ local optima, where random walker can get stuck.

Basic idea of Simulated Annealing





- Final point of the random walk: output (produced form).
- Grammatical, if final point is globally optimal.
- Otherwise, performance error.
- **Precision** of the algorithm depends on its speed (!!).
- Harmonic Serialism: same, but never move to a less optimal neighbour.

Overview

Example – Fast speech: Dutch metrical stress

fo.to.toe.stel 'camera'	*uit.ge.ve.rij* 'publisher'	*stu.die.toe.la.ge* 'study grant'	*per.fec.tio.nist* 'perfectionist'
susu	ssus	susuu	usus
fó.to.tòe.stel fast: *0.82* slow: *1.00*	*ùit.gè.ve.ríj* fast: *0.65* / **0.67** slow: *0.97* / **0.96**	*stú.die.tòe.la.ge* fast: *0.55* / **0.38** slow: *0.96* / **0.81**	*per.fèc.tio.níst* fast: *0.49* / **0.13** slow: *0.91* / **0.20**
fó.to.toe.stèl fast: *0.18* slow: *0.00*	*ùit.ge.ve.ríj* fast: *0.35* / **0.33** slow: *0.03* / **0.04**	*stú.die.toe.là.ge* fast: *0.45* / **0.62** slow: *0.04* / **0.19**	*pèr.fec.tio.níst* fast: *0.39* / **0.87** slow: *0.07* / **0.80**

Simulated / **observed** (Schreuder) frequencies.

In the simulations, $T_{step} = 3$ used for fast speech and $T_{step} = 0.1$ for slow speech.

Playing with the model

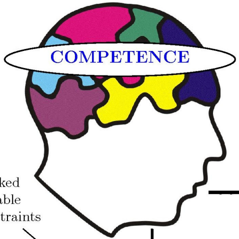
`http://www.birot.hu/sa-ot/index.php`

- Observe precisions as a function of speed.
- Play around with different parameter values.

Constraints: (`http://www.birot.hu/publications/BiroT-CLINproc2004.pdf`)

- ALIGN-LEFT: assign one violation mark if left edge of word does not align with left edge of some foot.
- OUTPUT-OUTPUT CORRESPONDENCE: the stress pattern matches the expectations from the morphological structure (z-parameter).
- $*\Sigma\Sigma$: one violation mark per adjacent feet borders.
- PARSE: one violation mark per unparsed syllable.
- TROCHAIC: one violation mark to each iambic foot ([us]).

Overview



A grammar is a Harmony function on the candidate set, defined by the ranked constraints.
Global optimum: more harmonic than all other candidates.
Local optimum: more harmonic than its neighbours.

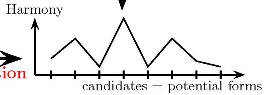
PERFORMANCE

Ranked violable constraints

Candidates (potential forms)

/aat/	NoCoda	Parse	Onset
☞ [a.a<t>]		*	**
[a.at]	*!		**
[<a>at]	*!	*	*

implementation



Optimality Theory
grammar
competence model
grammatical form = ☞ (globally) optimal candidate

SA-OT
implementation
performance model
produced forms = globally or locally optimal candidates

Conclusions

- Optimality Theory and Harmony Grammar: optimization problems.
- Not "constraints", rather elementary functions.
- Harmony: different ways of building a single target function from the elementary functions (OT vs. HG).
- Performance = implementation of the grammar.
- Simulated annealing: for instance normal vs. fast speech.
- Neighbourhood structure on the candidate set, local optima.

Thank you for your attention!

Tamás Biró:
`t.s.biro@uva.nl`

Work done and defended at / follow up (being) supported by:

university of groningen
NWO Netherlands Organisation for Scientific Research
University of Amsterdam