

Constraint Weighting as Linear Programming*

Rajesh Bhatt, Joe Pater and Christopher Potts
University of Massachusetts, Amherst

In Optimality Theory (OT: Prince and Smolensky 1993/2004), *ranking* is used to regulate constraint conflict: where two constraints differ in their assessment of output candidates, the higher ranked constraint determines which one is chosen:

(1) Optimality through ranking

Input	Constraint 1	Constraint 2
☞ Output 1		*
Output 2	*	

In OT's immediate predecessor Harmonic Grammar (HG: Smolensky *et al.* 1993, Legendre and Smolensky 2005), constraint *weighting* is used instead:

(2) Optimality through weighting

Weight	2	1	Σ
Input	Constraint 1	Constraint 2	
☞ Output 1		*	1
Output 2	*		2

Here we assume Prince's 2002 simplified version of HG; see also Keller 2005:

(3) The optimal candidate has the lowest sum of weighted violations (Σ)

Why ranking rather than weighting?

- (4) i. So phonologists don't have to do math (possibly apochryphal)
- ii. Because weighting produces implausible linguistic systems (Legendre, Sorace and Smolensky 2005; cf. Pater 2006)

* Acknowledgments: Thanks to Tim Beechey, Karen Jesney, John McCarthy and Paul Smolensky for very helpful discussion, as well as participants in Ling 730, UMass Amherst Fall 2005, especially Kathryn Flack and Shigeto Kawahara.

In this talk, we introduce an application of linear programming that:

- (5) i. Allows phonologists to use weighted constraints in phonological analysis without doing the sometimes genuinely complicated math
- ii. Allows for a more thorough assessment of the differences between systems with weighted and ranked constraints

1. Weighting implications

When constructing an analysis with weighted constraints, the goal is:

- (6) To find a set of weightings such that every optimal candidate has a lower Σ -value than each of its competitors

For the example in (2), the following *weighting implication* holds:

- (7) $w(\text{Con1}) > w(\text{Con2})$

It's not too hard to find values that satisfy this; let's look at a slightly more complicated case from natural language

Meccan Arabic, (Bakalla 1973, Abu-Mansour 1996, McCarthy 2003b) has a voicing contrast, but voicing assimilation fails to create voiced obstruents

Following the Local Conjunction analysis of similar phenomena (Bakovic 2000, Smolensky 2005), with weighted constraints we can analyze the blocking of assimilation to [+voice] as the combined effect of IDENT-VOICE and *VOICE (or equally, the assimilation to only [-voice] as due to *VOICE plus AGREE-VOICE)

(8) Constraint definitions

AGREE-VOICE: Adjacent obstruents have the same value for [voice]

IDENT-VOICE: Segments in correspondence have the same value for [voice]

*VOICE: Obstruents are [-voice]

- (9) Voiced obstruents are allowed: $w(\text{IDENT-VCE}) > w(*\text{VCE})$

/ʔibnu/	AGREE-VCE	IDENT-VCE	*VCE
☞ [ʔibnu]			1
[ʔipnu]		1	

(10) Voiced obstruents assimilate: $w(\text{AGREE-VCE}) + w(*\text{VCE}) > w(\text{IDENT-VCE})$

/ʔagsam/	AGREE-VCE	IDENT-VCE	*VCE
[ʔagsam]	1		1
☞ [ʔaksam]		1	

(11) Voiceless obstruents do not: $w(\text{IDENT-VCE}) + w(*\text{VCE}) > w(\text{AGREE-VCE})$

/ʔakbar/	AGREE-VCE	IDENT-VCE	*VCE
[ʔagbar]		1	2
☞ [ʔakbar]	1		1

The set of weighting implications:

- (12) $w(\text{IDENT-VCE}) > w(*\text{VCE})$
 $w(\text{AGREE-VCE}) + w(*\text{VCE}) > w(\text{IDENT-VCE})$
 $w(\text{IDENT-VCE}) + w(*\text{VCE}) > w(\text{AGREE-VCE})$

Solution?

2. From weighting implications to linear programming

Phonological analysis in OT is often much more complex than the example we have just looked at, involving many more constraints, many more inputs (URs), and many more candidate outputs (SRs)

The task of finding a suitable weighting by hand could therefore be a considerable obstacle to pursuing an analysis with weighted constraints. This leads to our first application of linear programming:

- (13) Allows phonologists to use weighted constraints in phonological analysis without doing the sometimes genuinely complicated math

The mathematician George Dantzig discovered the simplex algorithm in 1947, while working for the Air Force. He called it linear programming, where programming has an old military sense: troop movements, supply distribution, etc.

- (14) The tremendous power of the simplex method is a constant surprise to me.
 — George Dantzig

Linear programming optimizes a value, subject to a set of constraints

The value being optimized is expressed in an *objective function*, the set of constraints are expressed as a set of *inequalities*

In our implementation of Cormen *et al.*'s simplex algorithm, the objective function consists of the minimizations of the summed weights of the constraints, and the inequalities are transformed weighting implications

(15) Our objective function

Minimize $w_1 + \dots + w_n$ (where n = number of constraints)

We minimize the values because otherwise our systems would all be unbounded, and there would be no optimal solution

(16) Unboundedness of maximization

Given a set of weights $\{w_1 \dots w_n\}$ that satisfies the weighting implications, these values can all be multiplied by a constant resulting in a sum $w_1 + \dots w_n$ that is greater than the original one, and still satisfies the implications

Minimization is bounded because the constraints in linear programming are expressed in terms of inequalities (greater or equal to), rather than in terms of strict inequalities, and we impose a minimum value of 1 on all weights ($w_x \geq 1$).

The weighting implications are transformed as follows:

(17) Transformation of weighting implication to inequality

$$w_1 > w_2 \quad \rightarrow \quad w_1 - w_2 > 0 \quad \rightarrow \quad w_1 - w_2 \geq 1$$

In practice, it is possible to construct an inequality directly from the violation pattern of an optimal form and its paired sub-optimal candidate:

(18) Transformation of violation pattern to an inequality

Input	C1	C2
☞ Output 1		1
Output 2	1	

$$\rightarrow \quad wC1 - wC2 \geq 1$$

3. Weighted constraint analysis through a web interface

(19) <http://wwwx.oit.umass.edu/~linguist/potts/constraint-weighting/>

Phonological constraint weighting as linear programming

[Christopher Potts](#), [Joe Pater](#), and [Rajesh Bhatt](#)

Constraint weighting systems might be extremely promising for phonological analysis. But the process of finding weightings by hand is long, boring, and hard. This is a huge obstacle to fair assessment. We're here to make life easier. We show, [in the handouts below](#), that constraint weighting grammars reduce to linear programs, which are solvable using computationally efficient optimization algorithms. This page is our gateway into such an algorithm. You upload two Praat files. The program converts them into a linear system and solves it using the two-phase simplex method. The output is an HTML file displaying the minimal weighting if there is one, else a verdict of 'infeasible'. We are work on a typology calculator; the typological algorithm is described in the handout by Potts below.

We are presently testing the current implementation. Please write to [Chris](#) if you discover any bugs.

Minimal consistent weightings

<p>Please upload a data file for processing. <input type="button" value="Choose File"/> no file selected</p> <p>Please upload a violations file for processing. <input type="button" value="Choose File"/> no file selected</p> <p><input type="button" value="submit these files"/> <input type="button" value="clear"/></p>	<p>Sample files</p> <p>You can use these directly, and they illustrate the proper formatting.</p> <ul style="list-style-type: none">• Data file: hungarian-data.txt Violations file: hungarian-violations.txt• Data file: wolof-data.txt Violations file: wolof-violations.txt• Data file: inconsistent-data.txt Violations file: inconsistent-violations.txt
---	--

The file format we are using is the one Praat (Boersma and Weenick 2006) uses for OT grammar learning; this allows users to easily check learning outcomes with the Gradual Learning Algorithm (Boersma 1998)

Praat also supports the weighted constraint evaluation mode that we are employing - search the Praat manual for "LinearOT" (and see example 21)

The "data file" is a set of paired inputs and outputs. For the Meccan Arabic case, it looks like this:

(20) *Meccan Arabic data file*

```
"ooTextFile"  
"PairDistribution"  
3 pairs  
"?ibnu"    "?ibnu"    100  
"?agsam"   "?aksam"   100  
"?akbar"   "?akbar"   100
```

The associated numbers are used for learning distributions in Praat; they are irrelevant for our purposes.

The "violations file" is a set of candidate input-output mappings, with associated constraint violations:

(21) *Meccan Arabic violations file*

```
Object class = "OTGrammar 1"
```

```
<OptimalityTheory>  
3 constraints  
constraint [1]: "Agree-Vce" 100 100 ! Agree-Vce  
constraint [2]: "Ident-Vce" 100 100 ! Ident-Vce  
constraint [3]: "*Vce" 50 50 ! *Vce
```

```
0 fixed rankings
```

```
3 tableaus  
input [1]: "?ibnu" 2  
  candidate [1]: "?ibnu" 0 0 1  
  candidate [2]: "?ipnu" 0 1 0  
input [2]: "?agsam" 2  
  candidate [1]: "?agsam" 1 0 1  
  candidate [2]: "?aksam" 0 1 0  
input [3]: "?akbar" 2  
  candidate [1]: "?agbar" 0 1 2  
  candidate [2]: "?akbar" 1 0 1
```

Here is the output .html file for the Meccan Arabic example:

(22) Weightings found through linear programming

Optimal weighting and tableaux

Minimal feasible weighting for the data set		
Agree-Vce	Ident-Vce	*Vce
2	2	1

For each tableau, the winning output is the first listed.

Weight	2	2	1	
Input: ?agsam	Agree-Vce	Ident-Vce	*Vce	
?aksam	0	1	0	Weighted total: 2
?agsam	1	0	1	Weighted total: 3

Weight	2	2	1	
Input: ?akbar	Agree-Vce	Ident-Vce	*Vce	
?akbar	1	0	1	Weighted total: 3
?agbar	0	1	2	Weighted total: 4

Weight	2	2	1	
Input: ?ibnu	Agree-Vce	Ident-Vce	*Vce	
?ibnu	0	0	1	Weighted total: 1
?ipnu	0	1	0	Weighted total: 2

The "minimal" value refers to the sum of the weighting values. It can be no smaller than it is because we impose the following constraints:

- (23)
 - i. The weighted total for each winning output must be at least the weighted total of any loser plus 1
 - ii. The minimum value for any constraint is 1

We don't see this optimum as having any linguistic significance; this weighting is no more linguistically valuable than any other feasible weighting

It is, however, of some practical utility; the resulting values are usually small integers, and thus the math remains optimally simple. Also, minimization usually favors additive interaction between constraints, thus producing analyses differing from those of OT, which is useful in discovering differences between the theories.

4. Infeasibility and typology

For an analyst working with pen and paper, it could be particularly challenging to determine that no weighting whatsoever will render a set of output forms optimal relative to their competitors. Teasing apart possible from impossible sets of optima is the task when wants to determine the typological predictions of a set of constraints.

In OT, this is done by examining the results of all rankings of the constraint set; because the number of rankings is $n!$, where n is the number of constraints, this is termed factorial typology (Prince and Smolensky 1993/2004; see Hayes et al 2003 for a computational implementation)

With weighted constraints, the same strategy is infeasible, since the set of possible weightings is infinite, even for a single constraint (without imposing artificial restrictions on possible weighting values)

Linear programming is a particularly useful tool in this respect, since as well as being guaranteed to find the optimal solution, it will also label problems without solutions as *infeasible*

As a simple concrete example, consider the following seeming plausible outcome of additive weighted constraint interaction (cf. Prince 2002, Pater 2006):

(24) One coda is tolerated, but if there are two codas, one will be deleted

(25) NoCODA Every syllable ends in a vowel
 MAX Every input consonant has an output correspondent

We do not need linear programming to show that such a system is impossible under our assumptions, since the weighting implications are clearly inconsistent:

(26) Deletion of one of two codas: $w(\text{NoCODA}) > w(\text{MAX})$

/bantan/	NoCODA	MAX
ban.tan	2	0
☞ ba.tan	1	1

(27) Preservation of a single coda: $w(\text{MAX}) > w(\text{NoCODA})$

/batan/	NoCODA	MAX
☞ ba.tan	1	0
ba.ta	0	1

Morals:

- (28) i. Optimization systems do not impose a numeric cut-off on well-formedness; they seek the optimal form for a given input
 ii. NoCODA and MAX violations trade off one-to-one; there is no way of avoiding multiple NoCODA violations with a single MAX violation (cf. Meccan Arabic, where AGREE trades off against IDENT and *VCE)

For more complicated systems, however, linear programming yields an automatic assessment of feasibility

A fragment of Hungarian vowel harmony, as described by Hayes and Londe (to appear):

- (29) a. Backness harmony applies to suffixes, but not roots
 b. A root that ends in a neutral vowel preceded by a back vowel permits both back and front suffixes
 c. A root that ends in a neutral vowel preceded by a front vowel allows only a front suffix

Here is the data file with abstract forms corresponding to those generalizations (assumptions about URs are different from Hayes and Londe to appear):

(30) "Hungarian" data file (with relevant generalizations)

"BF"	"BF"	100	(28a)
"FB"	"FB"	100	(28a)
"B-nak"	"B-nak"	100	(28a)
"F-nak"	"F-nek"	100	(28a)

"FB-nak"	"FB-nak"	100	(28a)
"BF-nak"	"BF-nek"	100	(28a)
"BN-nak"	"BN-nak"	100	(28b)
"BN-nek"	"BN-nek"	100	(28b)
"FN-nak"	"FN-nek"	100	(28c)

(31) Constraint definitions (in the order of violations below)

- AGREE-B Adjacent vowels agree in backness
 *[BF], *[FB], *[B-nak], *[F-nak]...
- IDENT-B A segments' backness value is identical in Input and Output
 */BF/ → [BB], */F-nak/ → [F-nek]
- IDENT-B-ROOT A root segments' backness value is identical in Input and Output
 */BF/ → [BB]
- AGREE-DIST-B A back vowel is followed by only back vowels in the word
 *[BF], *[BN-nek]
- AGREE-LOC-NN A neutral vowel is followed by an adjacent front vowel
 *[BN-nak], *[FN-nek]
- AGREE-DIST-F A front vowel is followed by only front vowels
 *[FB], *[FN-nak]

(32) Portion of violations file:

```
input [1]: "BF" 2
  candidate [1]: "BB" 0 1 1 0 0 0
  candidate [2]: "BF" 1 0 0 1 0 0
input [5]: "F-nak" 3
  candidate [1]: "F-nak" 1 0 0 0 0 1
  candidate [2]: "F-nek" 0 1 0 0 0 0
  candidate [3]: "B-nak" 0 1 1 0 0 0
Input [7]: "BN-nak" 3
  candidate [1]: "BN-nak" 0 0 0 0 1 0
  candidate [2]: "BN-nek" 0 1 0 1 0 0
  candidate [3]: "BB-nak" 0 1 1 0 0 0
Input [8]: "BN-nek" 3
  candidate [1]: "BN-nak" 0 1 0 0 1 0
  candidate [2]: "BN-nek" 0 0 0 1 0 0
  candidate [3]: "NN-nek" 0 1 1 0 0 0
Input [9]: "FN-nak" 3
  candidate [1]: "FN-nak" 0 0 0 0 1 1
  candidate [2]: "FN-nek" 0 1 0 0 0 0
  candidate [3]: "BB-nak" 0 1 1 0 0 0
```

(33) Weighting discovered by linear programming:

Minimal feasible weighting for the data set						
Agree	Ident	Ident-B-Rt	Agree-Dist-B	Agree-Loc-NN	Agree-Dist-F	
3	1	4	1	1	1	

(34) Backness harmony applies to suffixes

Weight	3	1	4	1	1	1	
Input: F-nak	Agree	Ident	Ident-B-Rt	Agree-Dist-B	Agree-Loc-NN	Agree-Dist-F	
F-nek	0	1	0	0	0	0	Weighted total: 1
F-nak	1	0	0	0	0	1	Weighted total: 4
B-nak	0	1	1	0	0	0	Weighted total: 5

(35) Contrast after BN

Weight	3	1	4	1	1	1	
Input: BN-nak	Agree	Ident	Ident-B-Rt	Agree-Dist-B	Agree-Loc-NN	Agree-Dist-F	
BN-nak	0	0	0	0	1	0	Weighted total: 1
BN-nek	0	1	0	1	0	0	Weighted total: 2
BB-nak	0	1	1	0	0	0	Weighted total: 5

Weight	3	1	4	1	1	1	
Input: BN-nek	Agree	Ident	Ident-B-Rt	Agree-Dist-B	Agree-Loc-NN	Agree-Dist-F	
BN-nek	0	0	0	1	0	0	Weighted total: 1
BN-nak	0	1	0	0	1	0	Weighted total: 2
NN-nek	0	1	1	0	0	0	Weighted total: 5

(36) No contrast after FN

Weight	3	1	4	1	1	1	
Input: FN-nak	Agree	Ident	Ident-B-Rt	Agree-Dist-B	Agree-Loc-NN	Agree-Dist-F	
FN-nek	0	1	0	0	0	0	Weighted total: 1
FN-nak	0	0	0	0	1	1	Weighted total: 2
BB-nak	0	1	1	0	0	0	Weighted total: 5

A typological question:

(37) Will some weighting of this set of constraints produce harmony in roots, but not in suffixes?

(38) A portion of the relevant data file:

```

"BF"          "BB"    100
"FB"          "BB"    100
"B-nak"       "B-nak"   100
"B-nek"       "B-nek"   100
"F-nak"       "F-nak"   100
"F-nek"       "F-nek"   100
    
```

(39) The answer given by linear programming:

The system is infeasible.

Optimal weighting and tableaux

Minimal feasible weighting for the data set						
AGREE	IDENT	IDENT-B-RT	Agree-Dist-B	Agree-Loc-NN	Agree-Dist-F	
---	---	---	---	---	---	

This is consistent with what one can prove relatively straightforwardly by hand, given that there is no IDENT constraint specifically targeting suffixes, and no AGREE constraint specifically targeting roots.

5. Typological prospects

Legendre et al. (2005) show how the interaction of the following two constraints produces distinct results under ranking and weighting:

(40) ALIGN-HEAD-R

Assess one violation mark for every syllable intervening between the main stress and the right edge of the word

WEIGHT-TO-STRESS

Assess one violation mark for every unstressed heavy syllable

They point out that weighting can produce a system that stresses an initial heavy syllable that is followed by three light syllables, but that stresses the final syllable if more syllables follow the initial heavy:

(41) A problem for weighted constraints

Weight:	3.5	1	Σ
/bantana/	W-S	ALIGN-HD-R	
ban.ta.na.má	1	0	3.5
☞ bán.ta.na.ma	0	3	3

Weight:	3.5	1	Σ
/bantavama/	W-S	ALIGN-HD-R	
☞ ban.ta.na.va.má	1	0	3.5
bán.ta.na.va.ma	0	4	4

Stress systems are extremely well-studied typologically, and they do not display this sort of pattern. However McCarthy (2003a) shows that gradient alignment constraints like ALIGN-HEAD-R are also problematic in OT. An initial comparison of the results for weighting and ranking using other constraints indicates that the differences are far more subtle than might be suspected (Pater 2006)

Our expectation:

- (42) The availability of a tool for the automatic calculation of the typological predictions of weighted constraints will allow much deeper understanding of the relative power of ranked and weighted constraint systems

Ongoing work:

- (43) Automatic typology calculation from violations file; examines all sets of optima to determine which are feasible

- (44) Proof that the constraints we impose do not rule out feasible solutions

Sketch 1: given a solution in which some constraint has a weight x that is less than 1, the weights of all of the constraints in that solution can be multiplied by a constant $1/x$, such that x will equal 1

Sketch 2: given a solution in which the difference between the weighted total for a loser and a winner is less than 1, the weights can again be multiplied by a constant such that the difference is 1 or greater

- (45) Using objective function coefficients to introduce biases

6. Wider prospects

Further research aided by this implementation of linear programming should help to answer the following question:

- (46) Is a theory of phonology that replaces constraint ranking with weighting equally, or even better suited to model the range of phonological systems found cross-linguistically?

A positive answer to this question allows phonology to make closer contact with research in other domains, including phonetics (Flemming 2001), probabilistic learning (Goldwater and Johnson 2003, Hayes 2005, Wilson 2005, Jaeger to appear), and the mathematical study of optimization.

References

Abu-Mansour, Mahasen Hasan. 1996. Voice as a privative feature. Perspectives on Arabic Linguistics VIII: Papers from the Eighth Annual Symposium on Arabic Linguistics. Ed. Mushira Eid. Amsterdam and Philadelphia: John Benjamins.

Bakalla, Mohammed. 1973. The Morphology and Phonology of Meccan Arabic. Doctoral dissertation. SOAS, University of London.

Bakovic, Eric. 2000. Harmony, Dominance and Control. Ph.D. dissertation, Rutgers University.

Boersma, Paul. 1998. *Functional phonology: Formalizing the interactions between articulatory and perceptual drives*. Ph.D. dissertation, University of Amsterdam.

Boersma, Paul and David Weenick. 2006. Praat software version 4.4.24.

Cormen, Thomas H., Charles E. Leiserson, Ronald L. Rivest, and Cliff Stein. 2001. Introduction to Algorithms. Cambridge, MA: MIT Press, 2 ed.

Dantzig, George B. 1981/1982. Reminiscences about the origins of linear programming. Operations Research Letters 1(2):43–48.

Flemming, Edward. 2001. Scalar and categorical phenomena in a unified model of phonetics and phonology. Phonology 18. 7-44.

Goldwater, S., and M. Johnson. 2003. Learning OT constraint rankings using a maximum entropy model. Proceedings of Stockholm Workshop on Variation within Optimality Theory.

Hayes, B. 2005. A generative model of phonotactics. Lecture handout, University of Michigan.

Hayes, Bruce and Zsuzsa Czirák Londe. To appear. Stochastic Phonological Knowledge: The Case of Hungarian Vowel Harmony. In Phonology.

Hayes, Bruce, Bruce Tesar, and Kie Zuraw (2003) "OTSoft 2.1," software package, <http://www.linguistics.ucla.edu/people/hayes/otsoft/>

Jaeger, G. To appear. Maximum Entropy Models and Stochastic Optimality Theory, to appear in J. Grimshaw, et al. (eds.), Architectures, Rules, and Preferences: A Festschrift for Joan Bresnan, CSLI Publications, Stanford.

Keller, F. 2005. Linear Optimality Theory as a Model of Gradience in Grammar. In Fanselow *et al.* eds., *Gradience in Grammar: Generative Perspectives*. Oxford University Press.

Legendre, G., A. Sorace and P. Smolensky. 2005. The Optimality Theory – Harmonic Grammar Connection. In Smolensky and Legendre *eds.*

McCarthy, J. 2003a. OT constraints are categorical. *Phonology* 20, 75-138.

McCarthy, J. 2003b. Comparative markedness [short version]. *Theoretical Linguistics* 29. 1-51.

Pater, J. 2006. Additive Optimization and Phonological Typology. Handout from a talk presented at the MIT/UMass Phonology Workshop, 01/06 (available at <http://wwwx.oit.umass.edu/~linguist/potts/constraint-weighting/>)

Prince, A. 2002. Anything Goes. Ms, Rutgers University.

Prince, Alan, and Paul Smolensky. 1993/2004. Optimality Theory: Constraint interaction in generative grammar. Technical Report, Rutgers University and University of Colorado at Boulder, 1993. Revised version published by Blackwell, 2004.

Prince, Alan and Paul Smolensky. 1997. Optimality: From neural networks to universal grammar. *Science* 275:1604–1610.

Smolensky, P. 2005. Optimality in Phonology II: Harmonic Completeness, Local Constraint Conjunction, and Feature Domain Markedness. In Smolensky and Legendre (eds.)

Smolensky, P., G. Legendre, and Y. Miyata. 1993. Integrating connectionist and symbolic computation for the theory of language. *Current Science* 64: 381–391.

Smolensky, P. and G. Legendre (eds.). 2005. *The Harmonic Mind*. MIT Press.

Wilson, Colin. 2005. Learning Phonology with Substantive Bias: An Experimental and Computational Study of Velar Palatalization. Ms, UCLA.