OT grammars don't count, but make errors

The consequences of strict domination for simulated annealing[*]

Tamás Biró

ELTE Eötvös Loránd University, Budapest, Hungary

tamas.biro@btk.elte.hu, ORCID id: 0000-0002-8943-9276

http://www.birot.hu/, http://birot.web.elte.hu/

There were three greengrocers in Sziget street. Kardos, the first owner, put a sign in his window: "Best vegetables in town!". Then Kerekes, the owner of the second shop, raised the bid by posting: "Best vegetables of the world!" The third owner, Kohn, had a hard time. What should he do now? He finally decided to write on his door: "Best vegetables of the street!"

"Errare humanum est"—said the hedgehog when he climbed down the wire brush.

Abstract: Our goal is to compare *Optimality Theory* (OT) to *Harmonic Grammar* (HG) with respect to simulated annealing, a heuristic optimization algorithm. First, a few notes on Smolensky's ICS Architecture will bridge the gap between connectionist HG and symbolic HG. Subsequently, the latter is connected to OT via q-HG grammars, in which constraint C_i has weight q^i. We prove that q-HG converges to OT if $q \to +\infty$, even if constraint violations have no upper bound. This limit shall be referred to as the *strict domination limit.* Finally we argue that q-HG in the strict domination limit shares with OT a remarkable feature: simulated annealing does not always converge to 100% precision, even if the algorithm is offered ample time. Globally non-optimal local optima produced at slow pace will be viewed as *irregular forms.*

[*] Published online in: Beáta Gyuris, Katalin Mády and Gábor Recski (eds.), $K + K = 120$. *Papers dedicated to László Kálmán and András Kornai on the occasion of their 60th birthdays.* Research Institute for Linguistics, Hungarian Academy of Sciences, Budapest, 2017. ISBN: 978-963-9074-73-6. URL: http://www.nytud.hu/kk120/.
This research was supported by a *Marie Curie Career Integration Grant* within the *7th EU Framework Programme* (grant no. 631599, "MeMoLI"). The computations were run on the *Atlasz* HPC cluster of the Eötvös Loránd University. The very first draft of this paper grew out from a discussion with Reinhard Blutner in 2007. The author acknowledges the comments of an anonymous reviewer.

1 Do grammars count?

Well, we all are linguists, and so grammars do matter to us. And yet, it has become common wisdom in our profession that "grammars don't count". To quote Kornai (2008, p. 250): "the heavy emphasis on noncounting languages originates in an apocryphal remark of John von Neumann: *The brain does not use the language of mathematics.*"

This maxim has been used in several ways. Whether counting is legitimate or mistaken in language description, whether languages count segments, syllables or words (e.g., among many others, McCarthy, 2002; González, 2005; Watanabe, 2009; Graf, 2017), has been a long debate that shall not be our concern here. Our question is whether language *models* should make use of counting. More precisely: what is the consequence of the fact that *Optimality Theory* (Prince & Smolensky, 1993/2004) avoids counting, whereas *Harmonic Grammar* (Smolensky & Legendre, 2006) does count?

Even within Optimality Theory, 'to count or not to count' is a question raised multiple times. Apropos constraint violations, we all know that the violation level $C_k(x)$ assigned by constraint C_k to candidate x is usually a number of "stars". Yet, some constraints are simply categorical, binary, with range $\{0, 1\}$ or {`true`, `false`}, {`satisfies`, `violates`}: the last syllable of a word is either parsed into a foot or it is not, *Wh*-movement either has taken place or has not, the meaning is either faithfully expressed in the form or it is not, and so forth. Many other constraints are binary within some "locus", but the candidate itself is composed of several such "loci", and so they can be violated multiple times. These constraints *count* the number of marked segments or disfavored foot types or unfaithful features in the candidate. Some other constraints again may be violated to several degrees: for instance, the larger the distance of the head foot to some word edge (measured as the number of intervening syllables), the graver the violation of this alignment constraint by the candidate. Finally, some constraints can be gradually violated by several loci, and a non-trivial axiom of OT is that these constraints simply sum up the violations by the loci. Which of these constraints should and which should not be used is again a long story (McCarthy, 2002, 2003; Bíró, 2003; Eisner, 1997).

In standard OT, the counting by a constraint C_k is usually only a technicality, which boils down to the question which of $C_k(x)$ and $C_k(y)$ is *greater* (a more severe case of constraint violation). The specific numerical values actually only matter in Harmonic Grammar. Thus we arrive at the question that shall concern us here: is counting involved when constraints are combined into a single architecture?

Thus, given is a set $\{C_1, C_2, \ldots, C_n\}$ of constraints. Of these constraints, both *Harmonic Grammar* (HG) and *Optimality Theory* (OT) build up an objective function (target function) $H(x)$ to be optimized. While HG uses a weighted sum of the violations $C_k(x)$ [refer to equation (6) later], OT creates a vector, best known as the row corresponding to candidate x in an OT tableau [cf. (9)]. Both approaches postulate the output (e.g., surface form) SF(u) corresponding to input (e.g., underlying form) u to be the most harmonic element of the candidate set Gen(u):

$$\mathrm{SF}(u) = \underset{x \in \mathrm{Gen}(u)}{\arg\mathrm{opt}} \; H(x) \qquad (1)$$

In Harmonic Grammar, optimization is simply minimization in terms of the arithmetic *greater than* relation. Whereas in OT, it is the *lexicographic order* on a set of real-valued vectors: you compare the first components of the two vectors (the violations of the highest ranked constraint); if they are equal, then you proceed with comparing their second components; and so forth (e.g., Eisner, 2000; Jäger, 2002; Prince, 2002).

The output SF(u) can be conceived of as the *grammatical form*, that is, the form predicted by the grammar, the model of human *linguistic competence* (Newmeyer, 1983). The next step is actually to find the candidate x that optimizes the objective function $H(x)$, a procedure that has been compared to *linguistic performance* (Smolensky & Legendre, 2006; Bíró, 2006).

What procedure shall we use to find the optimal candidate? Similarly to the third greengrocer in Sziget street, we shall optimize locally. But as the hedgehog warns us, local optimization can go wrong. We discuss simulated annealing, a probabilistic hill climbing algorithm that performs local search, comparing its behavior with OT to its behavior with HG.

This article is structured as follows. Section 2 presents the link between connectionist Harmonic Grammar and symbolic Harmonic Grammar, also summarizing Smolensky's ICS Cognitive Architecture in passing. Subsequently, section 3 stretches the connection to Optimality Theory by introducing the concept of q-HG, a variant of Harmonic Grammar with exponential weights. As a new mathematical result, we show that q-HG converges to OT as the base of the exponents q grows infinite, even if no upper bound exists on the number $C_k(x)$ of violation marks. Then, section 4 introduces simulated annealing, before section 5 elaborates on why it works in most cases. In contrast to that, section 6 explains the main message of this paper: simulated annealing can fail in the *strict domination limit* ($q \to +\infty$). This point is illustrated by computer experiments in section 7, before drawing the conclusions in section 8.

2 From connectionist HG to symbolic HG

In order to understand why the optimization technique called *simulated annealing* is relevant for Optimality Theory, let us first recapitulate the connectionist idea behind OT. This section can also be read as an introduction to Paul Smolensky's *Integrated Connectionist/Symbolic Cognitive Architecture* (ICS) (Smolensky & Legendre, 2006).

A *Boltzmann machine* (Fig. 1) is a set of N nodes $\left(a_i\right)_{i=1}^{N}$, each with activation value a_i, a real number. It also includes, for each i and j, the connection strength W_{ij} of the arc from node i to node j, a real number again. Skipping some technical details often included in the literature on Boltzmann machines, let the *energy* E of the Boltzmann machine – the negative of the Harmony mentioned earlier and to be introduced soon – be simply the following sum of multiplications, over the connections i to j:

$$E = \sum_{i,j=1}^{N} a_i \cdot W_{ij} \cdot a_j. \qquad (2)$$

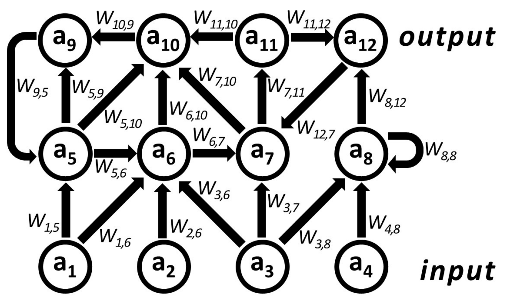
output
input
a_1 a_2 a_3 a_4 a_5 a_6 a_7 a_8 a_9 a_{10} a_{11} a_{12}
$W_{10,9}$ $W_{11,10}$ $W_{11,12}$ $W_{9,5}$ $W_{5,9}$ $W_{5,10}$ $W_{6,10}$ $W_{7,10}$ $W_{7,11}$ $W_{12,7}$ $W_{8,12}$ $W_{8,8}$
$W_{5,6}$ $W_{6,7}$ $W_{1,5}$ $W_{1,6}$ $W_{2,6}$ $W_{3,6}$ $W_{3,7}$ $W_{3,8}$ $W_{4,8}$

Figure 1: A *Boltzmann machine* with twelve nodes, an input layer and an output layer.

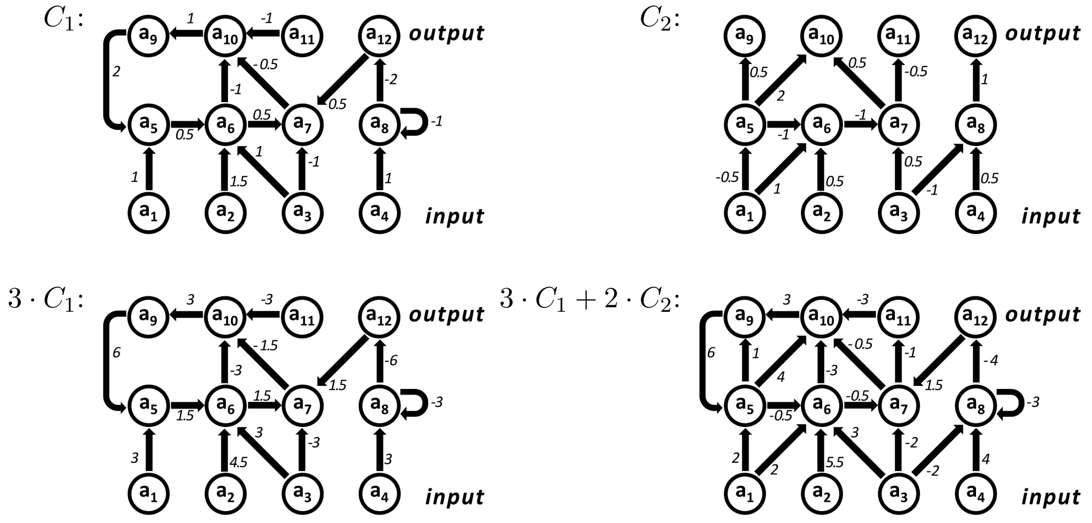

Figure 2: Examples of two constraints as partial connection strengths (upper row), and their weighted sums (linear combinations) (lower row). A missing arc means strength 0.

In *connectionist HG*, a *constraint* C_k is a set of partial connection strengths: some W^k_{ij} for each arc (i, j) in the network (Fig. 2). If the weight of C_k is w_k, and there are n constraints, then the total connection strength for arc (i, j) in the ensuing network is

$$W_{ij} = \sum_{k=1}^{n} w_k \cdot W^k_{ij}. \tag{3}$$

The notion of *candidate* in symbolic OT and HG, introduced in the previous section, corresponds to an activation pattern in connectionist HG. Some of the nodes describe the input (e.g., underlying representation), and are *clamped* (set) during computation. Some other nodes encode the output by the end of the computation. The rest of the nodes are "hidden". They may correspond to hidden (or covert) information, not present either in the input or in the output, but encoded in the candidate, as it plays some role in linguistic theory: syllable structure (Soderstrom, Mathis, & Smolensky, 2006), prosodic and syntactic parsing brackets, correspondence relations between input and output (cf. McCarthy & Prince, 1995), intermediate levels of representations (e.g., Boersma, 2011), and so forth.

During computation, the Boltzmann machine moves toward a (locally) minimal energy state, with its input nodes clamped and its output nodes eventually encoding the optimal output. The *candidate set* Gen(u) for input u is thus formed by the possible states of the network, with its input nodes fixed to encode u.

From (2) and (3), we get the *energy of a connectionist HG model* in state $A = \big(a_i\big)_{i=1}^{N}$:

$$E[A] = \sum_{i,j=1}^{N} a_i \cdot W_{ij} \cdot a_j = \sum_{i,j=1}^{N} a_i \cdot \sum_{k=1}^{n} w_k \cdot W_{ij}^k \cdot a_j = \sum_{k=1}^{n} w_k \cdot \sum_{i,j=1}^{N} a_i \cdot W_{ij}^k \cdot a_j. \tag{4}$$

We may now identify $C_k[A] = \sum_{i,j=1}^{N} a_i \cdot W_{ij}^k \cdot a_j$ as the *violation of constraint* C_k by the activation pattern (i.e., candidate) $A = \big(a_i\big)_{i=1}^{N}$. In turn,

$$E[A] = \sum_{k=1}^{n} w_k \cdot C_k[A]. \tag{5}$$

This is the equation that creates the bridge between connectionist HG and symbolic HG.

For instance, the violation of constraint C_1 in Fig. 2 is $a_1 \cdot a_5 + 0.5 \cdot a_5 \cdot a_6 + 1.5 \cdot a_2 \cdot a_6 + \ldots$ Similarly, the violation of C_2 turns to be $-0.5 \cdot a_1 \cdot a_5 - a_5 \cdot a_6 + a_1 \cdot a_6 + 0.5 \cdot a_2 \cdot a_6 + \ldots$ As a result, a connectionist harmonic grammar with weights $w_1 = 3$ and $w_2 = 2$ corresponds to the linear combination $2 \cdot a_1 \cdot a_5 - 0.5 \cdot a_5 \cdot a_6 + 2 \cdot a_1 \cdot a_6 + 5.5 \cdot a_2 \cdot a_6 + \ldots$

To summarize, a candidate x in symbolic HG corresponds to an activation pattern A of the Boltzmann network. A constraint C_k is a set of partial connection strengths W_{ij}^k, and its violation $C_k(x)$ by candidate x turns into the sum $\sum_{i,j=1}^{N} a_i \cdot W_{ij}^k \cdot a_j$. Hereby, the energy $E[A]$ of the Boltzmann network will map to the (negative) Harmony $H(x)$ of the HG grammar. Boltzmann machines, as a connectionist technique, minimize their energy, and so symbolic HG linguistic models also ought to optimize their harmony.

3 From symbolic HG to symbolic OT, via q-HG

To summarize, we have derived the connection between minimizing the energy $E[A]$ in connectionist HG and maximizing the harmony $H(x)$ in symbolic HG. The opposite directions of optimization can be taken care of with a negative sign, and it has historical reasons. To tell the truth, I personally prefer the minimization perspective even in OT, since the best candidate is the one that violates the constraints the least. We will nevertheless have to introduce that negative sign in order to maintain the view according to which the best candidate maximizes its harmony, given constraint weights w_k:

$$H(x) = -\sum_{k=1}^{n} w_k \cdot C_k(x) \tag{6}$$

/u/ $q = 3$ $q = 5$	C_2 $r_2 = 2$ $w_2 = 3^2 = 9$ $w_2 = 5^2 = 25$	C_1 $r_1 = 1$ $w_1 = 3^1 = 3$ $w_1 = 5^1 = 5$	3-HG	5-HG	OT
$[x]$		****	-12	☞ -20	☞
$[y]$	*		☞ -9	-25	

Table 1: Counting cumulativity: $[y]$ is more harmonic than $[x]$ if $q = 3$.

Now we proceed further towards *Optimality Theory.* It has become customary, especially in the literature on learning, to assign a real-valued *rank* r_k to each constraint C_k (Boersma, 1997; Boersma & Hayes, 2001). The higher its rank, the higher the constraint will be ranked in OT. The most direct connection between *weights* and *ranks* is identifying them: $w_k = r_k$ ('linear HG'). A less self-evident connection, *exponential HG* (Boersma & Pater, 2016 [2008]), has however been more frequently employed: $w_k = \exp(r_k)$, with some base larger than 1 (such as 2 or 10 or $e = 2.71\ldots$). Exponentiating the ranks has the advantage that learning will never produce negative weights (Pater, 2009), as well as that it also makes learning more efficient (cf. the inefficiency of learning linear HG, as demonstrated by Magri, 2016).

Earlier, I have introduced an approach called *q-Harmonic Grammar* in which $w_k = q^{r_k}$ (Biró, 2009). Having the value of $q > 1$ fixed, we may define a 2-HG grammar, a 10-HG grammar or a 1.23-HG grammar, if $q = 2$ or 10 or 1.23, respectively. But we can also change the value of q. The difference between exponential HG and q-HG is a question of perspective: the former sets the base of exponentiation, and considers it merely as a technical detail, whereas the latter views it as an interesting tunable parameter.

On the one hand, a change of the basis q from q_1 to q_2 is technically equivalent to multiplying all ranks by the factor $\frac{\log q_2}{\log q_1}$. On the other hand, if the ranks are kept fixed, then increasing q is how we can get HG to turn into OT: we are demonstrating momentarily that under certain conditions an OT grammar and a q-HG grammar with the same constraint ranks define the same language in the $q \to +\infty$ limit.

Parameter q becoming infinitely large will be called the *strict domination limit.* The motivation of the expression is that the key difference between HG and OT is *strict domination*: if the two approaches predict different language typologies, then it is because HG, but not standard OT, allows *counting cumulativity* and *ganging-up cumulativity* (Jäger & Rosenbach, 2006).

Tableaux 1 and 2 illustrate the point. The best candidates, shown by the pointing hand, are calculated with respect to the hierarchy $(C_3 \gg) C_2 \gg C_1$ in OT; and as the weighted sum of the violations in q-HG, the weights being $w_i = q^{r_i}$. In both tableaux, candidate $[x]$ is more harmonic than $[y]$ for OT and 5-HG. However, for 3-HG, tableau 1 shows how multiple violations of the lower ranked constraint C_1 can turn the candidate $[y]$ more harmonic. Similarly, in tableau 2, the lower ranked two constraints, C_2 and C_3, gang up: while neither of them alone could make $[x]$ worse than $[y]$, taking them together results in $[y]$ winning over $[x]$.

/u/	C_3	C_2	C_1	3-HG	5-HG	OT
	$r_3 = 3$	$r_2 = 2$	$r_1 = 1$			
$q = 3$	$w_3 = 27$	$w_2 = 9$	$w_1 = 3$			
$q = 5$	$w_3 = 125$	$w_2 = 25$	$w_1 = 5$			
[x]		**	****	-30	☞ -70	☞
[y]	*			☞-27	-125	

Table 2: Ganging-up cumulativity: [y] is more harmonic than [x] if $q = 3$.

In both cases, 5-HG behaves like OT, and any q-HG would do so if $q \geq 5$. It has been long known (Prince & Smolensky, 1993/2004, p. 236) that a sufficient criterion for a harmonic grammar with an exponential weight system to display OT-like behavior is that the base of the exponential weights be not less than the highest amount of stars in a cell (which is 4 in our example) plus 1. This is why 5-HG is equivalent to OT, but not 3-HG. (For a reversed approach, refer to Prince, 2002.)

Let us now formalize this observation. $\mathcal{U}$ shall be the set of underlying forms – the domain of the universal Gen function – which is postulated to be universal by the *Richness of the Base* principle (Prince & Smolensky, 1993/2004, p. 225). Moreover, let us posit that our n constraints take non-negative integer values: for $k = 1, \ldots, n$, the constraint C_k is a mapping from $\bigcup_{u \in \mathcal{U}} \mathrm{Gen}(u)$ to $\mathbb{N}_0$. This last requirement will play a crucial role in the proof to be presented. While it certainly applies to most linguistic models in the OT and HG literature, it poses some limitations to the generalizability of the framework.

Without loss of generality, we can assume that the indices of the constraints reflect their ranking. Consequently, our OT grammar shall be

$$C_n \gg C_{n-1} \gg \ldots \gg C_1 \tag{7}$$

This OT grammar can be matched to the q-HG grammar with $r_k = k$ and $w_k = q^k$ (remember that $q > 1$). The point of interest is whether these two grammars generate the same language. Put it differently, the following two Harmony functions are compared:

$$H_q(x) = -\sum_{k=1}^{n} q^k \cdot C_k(x) \tag{8}$$

$$H_{\mathrm{OT}}(x) = \Big(- C_n(x), -C_{n-1}(x), \ldots, -C_1(x) \Big) \tag{9}$$

Equation (1), reformulated here, defines a grammar for either kind of Harmony functions:

$$\mathrm{SF}(u) = \underset{x \in \mathrm{Gen}(u)}{\arg\max}\ H(x) \tag{10}$$

(for all $u \in \mathcal{U}$). Such a grammar maps an underlying form u to a surface form s, if and only if $H(s) \succeq H(x)$ for all $x \in \mathrm{Gen}(u)$. SF(u) is the set of these optimal candidates. In the case of q-HG, the values of H_q are compared using the arithmetic *greater than* *or equal to* relation $\geq$, whereas in OT, the *lexicographic order* $\succeq_{\text{lex}}$ compares the H_{OT} vectors. In the former case, the set of optimal candidates will be denoted as $\text{SF}_q(u)$, and in the latter case, as $\text{SF}_{\text{OT}}(u)$

We now prove a theorem that guarantees that the OT grammar (7) and the corresponding q-HG grammar (8) map any $u \in \mathcal{U}$ to the same surface form(s), if q is sufficiently large. This fact has been long known (Prince & Smolensky, 1993/2004, p. 236), but only if the number of violations admitted by the constraints were limited. We now show that no such upper limit is required, if the constraints take integer values.

Theorem 1. *Given are non-negative integer constraints $C_n, C_{n-1}, \ldots, C_1$ (ordered by their indices) and a Generator function* Gen. *Then, for any underlying form $u \in \mathcal{U}$ there exists some threshold $q_0 \geq 1$ such that for all $q > q_0$, $\text{SF}_{OT}(u) = \text{SF}_q(u)$.*

Proof. For any given $u \in \mathcal{U}$, we shall construct such a q_0. In this proof, the symbols s, s_1, s_2 and x will always denote elements of $\text{Gen}(u)$.

First, observe that if $s_1 \in \text{SF}_{\text{OT}}(u)$ and $s_2 \in \text{SF}_{\text{OT}}(u)$, then from the definition of the optimal set $\text{SF}_{\text{OT}}(u)$, we obtain $H_{\text{OT}}(s_1) \succeq_{\text{lex}} H_{\text{OT}}(s_2)$ and $H_{\text{OT}}(s_2) \succeq_{\text{lex}} H_{\text{OT}}(s_1)$; from which it follows that they share the same violation profile. That is, they violate each constraint to the same level: $C_k(s_1) = C_k(s_2)$ for all k.

In turn, it is well-founded to introduce the threshold q_0 as

$$q_0 = 1 + \max\left\{C_k(s), C_{k-1}(s), \ldots, C_1(s)\right\}$$

for whichever $s \in \text{SF}_{\text{OT}}(u)$. Since the constraints are postulated to have a non-negative range, $q_0 \geq 1$ follows. Now we have to show $\text{SF}_{\text{OT}}(u) = \text{SF}_q(u)$ to hold for all $q > q_0$.

If $s_1 \in \text{SF}_{\text{OT}}(u)$ and $s_2 \in \text{SF}_{\text{OT}}(u)$, then they violate each constraint to the same level, and so $H_q(s_1) = H_q(s_2)$, for any q. In order to complete our proof, it remains to be shown that if $q > q_0$, $s \in \text{SF}_{\text{OT}}(u)$ and $x \notin \text{SF}_{\text{OT}}(u)$, then $H_q(s) > H_q(x)$. Candidates that are suboptimal for H_{OT} are also suboptimal for H_q.

Since $s \in \text{SF}_{\text{OT}}(u)$ and $x \notin \text{SF}_{\text{OT}}(u)$, the vector $H_{\text{OT}}(s)$ is strictly lexicographically greater than the vector $H_{\text{OT}}(x)$. This means that there exists some "fatal constraint" C_f such that for all $k > f$, $C_k(s) = C_k(x)$, and $-C_f(s) > -C_f(x)$. Since our constraints take integer values, we conclude that $C_f(s) - C_f(x) \leq -1$.

Moreover, observe that for any k, $C_k(s) - C_k(x) < q-1$. This inequality holds because by the above definition of q_0, $C_k(s) \leq q_0 - 1 < q - 1$, whereas by the non-negativity of all constraints, $C_k(x) \geq 0$.

These two inequalities on the differences of the violations yield, for all $q > q_0 \geq 1$,

$$\begin{aligned}
H_q(x) - H_q(s) &= \sum_{k=1}^{n} \big[C_k(s) - C_k(x)\big] \cdot q^k = \\
&= \big[C_f(s) - C_f(x)\big] \cdot q^f + \sum_{k=1}^{f-1} \big[C_k(s) - C_k(x)\big] \cdot q^k < \\
&< -1 \cdot q^f + \sum_{k=1}^{f-1} (q-1) \cdot q^k = -q^f + (q-1) \cdot \sum_{k=1}^{f-1} q^k = \\
&= -q^f + (q-1) \cdot \frac{q^f - q}{q-1} = -q < 0.
\end{aligned}$$

That is, $H_q(s) > H_q(x)$ indeed holds. To summarize, for each $u \in \mathcal{U}$, we have proposed a $q_0 \geq 1$ such that for all $q > q_0$, the elements of $\mathrm{SF_{OT}}(u)$ are equally harmonic in q-HG; but they are more harmonic with respect to H_q than the candidates *not* in $\mathrm{SF_{OT}}(u)$. Thus, OT and q-HG map u to the same optimal subset $\mathrm{SF}_q(u) = \mathrm{SF_{OT}}(u) \subseteq \mathrm{Gen}(u)$. □

Obviously, nothing requires that a single q_0 work for all elements of $\mathcal{U}$; rather q_0 is dependent on u. But as q grows, more and more underlying forms will be mapped to the same surface forms by OT and by q-HG. Let $q_{0(u)}$ be some threshold q_0 for u, such as the one constructed in the proof of Theorem 1. Since $\mathcal{U}$ is most often a countable set, we can sort its elements by $q_{0(u)}$. Let the $q_{0(u)}$ value of the kth element of $\mathcal{U}$ in this list be $q_{0[k]}$. Now, if you wish your q-HG grammar to map at least k elements of $\mathcal{U}$ to the same output as the corresponding OT grammar does, then you should have $q > q_{0[k]}$.

As an example, remember tableaux 1 and 2. We have seen that $q_0 = 5$ is a good threshold: for all $q > q_0 = 5$, the q-HG grammar corresponding to the OT grammar will yield the output that is also most harmonic in the OT approach. But imagine now a different input, $/u'/$, whose OT winner $[x']$ incurs 6 violations by constraint C_1. This second input will require $q_{0(u')} = 7$, a higher threshold. And yet, you can set q to 7.1, and your q-HG grammar turns equivalent to OT for both inputs. And so forth. Even if you do not have an *a priori* upper bound of the number of stars assigned by C_1, and even if you do not want to restrict the input set arbitrarily, you will know: whenever you are about to compute the most harmonic element of a candidate set, you can have a value of q such that q-HG may be used instead of OT.

If a *language* is the way it maps inputs (underlying forms) onto outputs (surface forms), then the functions (set of mappings) $\mathrm{SF_{OT}}$ and SF_q are simply the languages generated by an OT grammar and by a q-HG grammar, respectively. Alternatively, the Chomskyan *E-languages* would be the ranges of $\mathrm{SF_{OT}}$ and of SF_q, respectively.

The theorem just proven can be reformulated as follows: the language generated by q-HG converges to the language generated by OT, as q grows infinitely large; that is,

Corollary 2.

$$\lim_{q \to +\infty} \mathrm{SF}_q = \mathrm{SF}_{OT} \quad \text{pointwise.}$$

Here the *pointwise convergence* of a sequence of functions on $\mathcal{U}$ is understood as follows: for any $u \in \mathcal{U}$ there exists some q_0 such that for all $q > q_0$, $\mathrm{SF}_q(u) = \mathrm{SF}_{\mathrm{OT}}(u)$. The limit $q \to +\infty$ has been called the *strict domination limit* (Biró, 2009).

Before proceeding, a remark is in order. The proof crucially relied on the constraints taking non-negative integer values. In the general case, however, Corollary 2 might still hold, even if in a weaker sense.

Take the following HG and OT grammars: candidates are non-negative real numbers ($\mathrm{Gen}(u) = \mathbb{R}_0^+$), while the two constraints are $C_2(x) = (x-1)^2$ and $C_1(x) = x$. In OT, the single best candidate for the highest ranked constraint C_2 is $x^*_{\mathrm{OT}} = 1$. All other candidates incur more violations by C_2, and so C_1 plays no role. In q-HG, however, $H_q(x) = -q^2 \cdot (x-1)^2 - q \cdot x$, which takes its maximum at $x^*_q = \frac{2q-1}{2q}$. For no real q will $x^*_q = x^*_{\mathrm{OT}}$; and so no q_0 exists such that $\mathrm{SF}_q(u) = \mathrm{SF}_{\mathrm{OT}}(u)$ for all $q > q_0$.

Observe, though, that $\lim_{q \to +\infty} x^*_q = x^*_{\mathrm{OT}}$. In a weaker sense, Corollary 2 still holds, at least for this specific example: for all $u \in \mathcal{U}$ and all $\epsilon > 0$, there exists some q_0 such that for all $q > q_0$, the distance of $\mathrm{SF}_q(u)$ and $\mathrm{SF}_{\mathrm{OT}}(u)$ is less than ϵ. Readers worried about the linguistic relevance of this example should note that the factorial typology includes candidates 1 and 0, and so it can be seen as a model of how a continuous phonetic feature maps to categorical phonology: it is either present or absent from a language. And yet, for candidates that are symbols or objects without a meaningful distance metric, it would be hard to formulate a similar conjecture of convergence.

4 Simulated annealing for symbolic Harmonic Grammars

Once we have defined how our grammars map an input (or underlying form) onto an output (or surface form) as an optimum defined by eq. (1) or eq. (10), in the second half of this paper we turn to the next question: how to find this optimum? The Boltzmann machines underlying connectionist HG immediately come with an answer: *simulated annealing.*

In the case of symbolic OT, the answer may be much less obvious, and even 'hard'. While most of our colleagues happily rely on their intuitions, Lauri Karttunen (2006) demonstrated "the insufficiency of paper-and-pencil linguistics", arguing for finite-state implementations of OT. Finite-state OT, however, imposes requirements that are met by many, but not all linguistic models (cf. e.g., Eisner, 1997, Jäger, 2002 and Bíró, 2003, and references therein). Further approaches include dynamic programing (or chart parsing; Tesar & Smolensky, 2000) and genetic algorithms (Turkel, 1994; Pulleyblank & Turkel, 2000). It used to be a consensus in the field that the generation problem of OT is NP-hard in the size of the grammar (e.g., Eisner, 1997, 2000; Idsardi, 2006a, 2006b). This consensus was challenged by András Kornai in two squibs (2006a, 2006b) that probably made one of the liveliest moments in the history of the *Optimality List* and the ROA *Rutgers Optimality Archive* (and see also Heinz, Kobele, & Riggle, 2009).

Heuristic optimization algorithms, including simulated annealing and genetic algorithms, have been successfully deployed to find an approximately good solution for NP-hard problems (Reeves, 1995, pp. 6–11). While they do not guarantee to always return

```
ALGORITHM Gradient Ascent: OT with restricted GEN
 x := x_init;
 repeat
        x_prev := x;
        x      := most_harmonic_element( {x_prev} U neighbors(x_prev) );
 until x = x_prev
 return x                        # x is an approximation to the optimal solution
```

Figure 3: Gradient Ascent: iterated Optimality Theory with a restricted GEN (Do-α).

the best solution, they do so reasonably well, returning the optimum pretty often, and otherwise returning a solution almost as good as the best one. Whether the generation problem in OT is NP-hard, or it is not, two further arguments can also be given for the use of heuristic optimization: similar trends in the cognitive sciences in general, beyond linguistics (e.g., Gigerenzer, Todd, & the ABC Research Group, 1999), and the very fact that our human speech production is also known to be prone to errors. Hence our interest in “less perfect” approaches and the motivation to employ *simulated annealing* for Optimality Theory (Bíró, 2005a, 2005b, 2006).

Let me now summarize simulated annealing (in a way that is based on Biró, 2007). Equations (1) and (10) define Optimality Theory as an optimisation problem. The task is to find the candidate x^* that optimizes $H(x)$.

Many heuristic algorithms do not always find the (globally) optimal candidate, but are simple and still efficient because they exploit the structure of the search space, which is the candidate set in our case. This structure is realized by a *neighborhood relation*: for each candidate x there exists a set neighbors(x), the set of the neighbors of x. It is often supposed that neighbors differ only *minimally*, whatever that means. The neighborhood relation is usually symmetric, irreflexive and results in a connected graph-like structure: any two candidates are connected by a finite chain of neighbors. More details of this relation should depend on the specific linguistic phenomenon under discussion.

The neighborhood structure – also called the *topology* – invites for a *random walk* in the search space, that is, on the candidate set. This walk can be conceived of as a series $x_0, x_1, x_2, \ldots, x_L$ of candidates. Candidate x_i, to be also referred to as the position of the random walker at time i, must be either identical to, or a neighbor of the candidate x_{i-1}, the previous position of the random walker. Position x_0 will be called the *initial position* (x_{init}), and x_L shall be the *final position* (x_{final}) of the random walk, whose length is L, the number of “steps”.

A random walker, such as a hedgehog, will walk in a *landscape.* The landscape's horizontal map is provided by the neighborhood structure, whereas its vertical dimension is the objective function H to be optimized. The hedgehog's goal is to climb the highest point in this landscape.

The simplest algorithm, *gradient ascent*, comes in two flavors. The version on Fig. 3 defines x_{i+1} as the best element of the set $\{x_i\} \cup$neighbors(x_i). The hedgehog walks as long as x_{i+1} differs from x_i, and the algorithm is deterministic for each x_{init}. This kind of optimization has been known in Optimality Theory since 1993 (Prince & Smolensky,

```
ALGORITHM Randomized Gradient Ascent
 x := x_init ;
 repeat
          Randomly select x’ from the set neighbors(x);
          if  (x’ not less harmonic than x)    then   x := x’;
 until stopping condition = true
 return x                       # x is an approximation to the optimal solution
```

Figure 4: Randomized Gradient Ascent

1993/2004) as *serial evaluation* (McCarthy, 2007) or *harmonic serialism* (McCarthy, 2010): x_{init} is the underlying form, Do-α (a restricted version of Gen) creates the set $\{x\} \cup \texttt{neighbors}(x)$, whereas the Eval module finds its best element in each iteration.

The second version of *gradient ascent* is stochastic (Figure 4). In step i, the hedgehog chooses a random $x' \in \texttt{neighbors}(x_i)$, using some pre-defined probability distribution on this set (often a uniform distribution). If neighbor x' is not worse than x_i, then the next element x_{i+1} of the random walk will be x'; otherwise, x_{i+1} is x_i. The stopping condition requires the number of iterations to reach some sufficiently large value, or the average improvement of the objective function in the last few steps to drop below a threshold (usually zero). Then the algorithm returns the output x_{final}, which is likely to be a local optimum.

Simulated annealing (Fig. 5) plays with this second theme to increase the hedgehog's chances of finding the global optimum and avoid being trapped in unwanted local optima. The idea is the same, but if x' is worse than x_i, then there is still a chance to move to x'. Importantly, however, this probability is reduced to 0, as the algorithm proceeds. (In some versions of simulated annealing, which we ignore here, if x' is better than x_i, the chance of moving to x' is less than 1, with this probability gradually converging to 1.)

The *transition probability* of moving to x' depends on the objective function H at points x_i and x', as well as on a parameter of the algorithm, $T > 0$, called *temperature* for historical reasons (Metropolis, Rosenbluth, Rosenbluth, Teller, & Teller, 1953; Kirkpatrick, Jr., & Vecchi, 1983; Cerny, 1985):

$$P(x_i \to x'|T) = e^{\frac{H(x') - H(x_i)}{T}}. \quad (11)$$

(Note that usually an energy function E is minimized, and not a harmony function H maximized, and therefore the standard formula also includes a negative sign in the exponent.) If the randomly chosen neighbor x' is less harmonic than x (if $H(x') < H(x)$), then a random number r is generated, and we move to x' if and only if $r < P(x_i \to x'|T)$.

Temperature T is gradually decreased following a *cooling schedule*, a decreasing series of values for T, so that in step i, the value of the temperature is T_i:

$$T_{max} = T_0 > T_1 > T_2 > \ldots > T_i > \ldots > T_L = T_{min} > \text{ but close to 0.} \quad (12)$$

Some allow the same T_i value to be re-employed a finite number of times *rep*, independent of (Reeves, 1995, p. 26), or dependent on (Henderson, Jacobson, & Johnson, 2003) T_i.

```
ALGORITHM: Simulated Annealing
 Parameters:  x_init      # initial state (often randomly chosen)
              T_max       # initial temperature > 0
              alpha       # temperature reduction function = cooling schedule

 x := x_init ;
 T := T_max  ;
 Repeat
     Randomly select x' from the set neighbors(x);
     Delta :=  H(x') - H(x) ;
     if ( Delta > 0 )     # neighbor is more harmonic than current position
        then
            x := x' ;
        else
            # move to x' with transition probability P(Delta;T) = exp(Delta/T):
            generate random r uniformly in range (0,1) ;
            if ( r < exp ( Delta / T ) )
                  then    x := x' ;
            end-fi
     end-fi
     T := alpha(T)          # decrease T according to cooling schedule
 Until stopping condition = true
 Return w                  # w is an approximation to the optimal solution
```

Figure 5: *Maximizing* a real-valued harmony function $H(x)$ with simulated annealing.

As the temperature T decreases, the exponent in (11) becomes an increasingly low negative number, and the transition probability $P(x_i \to x'|T)$ converges to zero. With very low temperatures, the hedgehog would not move towards lower harmony anymore.

The final position of the random walker will be returned as the output of simulated annealing. It is a stochastic algorithm without the guarantee of always finding the global optimum. In fact, the hedgehog may be stuck in a local optimum, having climbed a hill that is not the highest in the landscape, but the temperature being too low already for the hedgehog to take a counter-optimal step. The probability of returning the global optimum will be referred to as the *precision* of the algorithm.

An important fact about simulated annealing is that precision can be made to converge to 1 as the number of iterations L grows (Reeves, 1995; Henderson et al., 2003). The next section illustrates why this happens so. Smolensky and Legendre (2006) repeatedly refer to this fact as an advantage of their proposal: if the language model is given ample time, it will almost certainly find the most harmonic, that is, the grammatical form. (Note, though, that both the formal analysis and the practice of simulated annealing know of cases that require very long running times; cf. Reeves, 1995, p. 63.) Bíró (2006) takes a different approach: the language model unable to find the grammatical form makes a performance error, similarly to the human brain. Moreover, if the algorithm is given less time, it makes more errors, like a fast speaking human. Speed can be traded for precision, and fast speech can be modeled with simulated annealing (Bíró, 2005a).

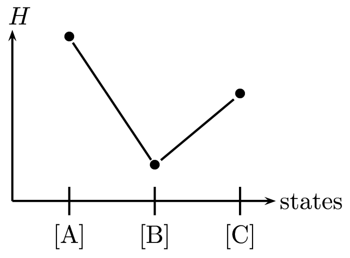

Figure 6: V landscape, with three candidates in a row. [B] in the center is worse than the two candidates on the peripheries. [A] is the global optimum, while [C] is locally optimal.

	C_2	C_1
☞ [A]		
[B]	*	
∼ [C]		*

Table 3: A possible tableau for the asymetrical V shaped landscape. Note symbol ∼ marking the local optimum.

So far, we have only discussed simulated annealing with a real-valued harmony function. However, I have earlier proposed a way to adopt simulated annealing (Fig. 5) to OT; in particular, I had to adapt the exponential expression in (11) to the non-real valued objective function in (9) (Bíró, 2005a, 2005b, 2006). While I have presented various mathematical arguments that lead to the same *Simulated Annealing for Optimality Theory* (SA-OT) *Algorithm* (Bíró, 2006, chapters 2 and 3; and a different train of thought in Biró, 2013 that could also be applied to SA-OT), critics may still argue that the SA-OT Algorithm is removed from "real" simulated annealing.

In particular, SA-OT lacks the above mentioned property of simulated annealing: in some cases increasing the number of iterations did not improve the precision of the algorithm (refer to Bíró, 2006, sections 2.3.2 and 6.4, as well as Biró, 2009). It has been argued that the reason is *strict domination.* Look at the landscape on Fig. 6 with tableau 3. This toy grammar includes two local optima separated by the suboptimal candidate [B]. [A] is the global optimum, better than [C] due to low ranked constraint C_1. However, in simulated annealing, non-adjacent candidates are not compared directly, and their frequencies in production emerge as a consequence of the landscape. (This is a significant difference between SA-OT and Maximum Entropy OT, cf. Goldwater & Johnson, 2003.) Both [A] and [C] defeat [B] by the highly ranked constraint C_2, and strict domination requires that lower ranked constraints do not play any role. Consequently, as shown in Bíró (2006, section 2.3.2), SA-OT will return both [A] and [C] with a probability of 50%, independently of the speed of the algorithm.

Without entering further details of the SA-OT Algorithm, the main thrust of the current paper is to explain that this non-convergence property is intrinsic to Optimality Theory: it appears not only in the SA-OT Algorithm, which adopts strict domination *a priori*, but also in the strict domination limit of applying simulated annealing to symbolic HG. In other words, strict domination grammars, which practically do not count, will make errors, even if ample time is available for them to perform computations.

5 Why does simulated annealing work?

5.1 A simple model and its dynamics

First, let us try to understand why standard simulated annealing is successful as an optimization algorithm: why its precision converges to 100% if the length of the random walk is increased. The key will be that in a certain phase of the algorithm (the "second phase") hedgehogs can escape local optima and be attracted to the global optimum.

Let us return to the simplest search space with a global optimum and another local optimum (Fig. 6), which will be referred to henceforth as an 'asymmetric V landscape'. Our randomly walking hedgehog in state [B] has two neighbors to choose from. Suppose that both neighbors have a chance of 0.5 to be picked. Then, either [A] or [C] is chosen, and the hedgehog will move there with a transition probability $P(B \to A, C|T) = 1$, because $H(B)$ is lower than both $H(A)$ and $H(C)$. If the random walker is, however, in state [A] or [C], the only neighbor has a reduced probability of being moved to. Following eq. (11), the chances are:

$$\begin{aligned} p_A(T) := P(A \to B|T) &= e^{\frac{H(B)-H(A)}{T}} \\ p_C(T) := P(C \to B|T) &= e^{\frac{H(B)-H(C)}{T}} \end{aligned} \tag{13}$$

Since $H(A) > H(C)$, we can easily see that $p_A(T) < p_C(T)$ at any time, at any temperature T. As a side remark, this inequality does not hold in SA-OT in the case presented by tableau 3, and that is why the global optimum [A] is not able to attract the random walker away from the other local optimum. Hence, the 50% precision of SA-OT.

Now suppose that many hedgehogs walk simultaneously. The average number of hedgehogs moving from state x to a neighboring state x' in some iteration of the algorithm is the product of the number of hedgehogs in state x, the probability of choosing neighbor x' when at x, and the transition probability $P(x \to x'|T)$. At time step i, let a_i, b_i and c_i denote the number of random walkers in states [A], [B] and [C] respectively. By using the above probabilities to calculate the flows from and into each state, we obtain:

$$\begin{aligned} a_{i+1} &= a_i - a_i \cdot p_A(T_i) + \frac{1}{2}b_i \\ b_{i+1} &= b_i - \frac{1}{2}b_i + a_i \cdot p_A(T_i) - \frac{1}{2}b_i + c_i \cdot p_C(T_i) \\ c_{i+1} &= c_i - c_i \cdot p_C(T_i) + \frac{1}{2}b_i \end{aligned} \tag{14}$$

As expected, the number of hedgehogs $a_i + b_i + c_i = N$ is constant in time. The precision of the algorithm is the probability that a random walker finishes the walk in the globally optimal state [A], which is a_∞/N. In what follows we discuss the mechanisms that increase this precision.

5.2 Extreme temperatures, and in between

Let us now consider two extreme cases. First, we suppose that temperature is very low from time step $i = 0$ onwards, that is, $T \ll H(C) - H(B)$. Then, according to (13), $p_A \approx 0$ and $p_C \approx 0$. In this case, the poor hedgehogs are unable to escape from the local optima. The initial population b_0 in state [B] is distributed within one step between the two local optima, and from this point onwards $a_i = a_0 + 0.5b_0$ and $c_i = c_0 + 0.5b_0$ for all $i > 0$. In brief, the random walkers are frozen into the local optima at very low temperatures.

In the second extreme case, temperature is very high: $T \gg H(A) - H(B)$, resulting in $p_A \approx 1$ and $p_C \approx 1$, from time step $i = 0$ onwards. Then the random walkers oscillate between the central and the peripheral positions. Since the random walkers are equally distributed by position [B], the whole system itself oscillates between two states. At odd time steps $a_{2i-1} = c_{2i-1} = 0.5b_0$ and $b_{2i-1} = a_0 + c_0$, whereas at even time steps $a_{2i} = c_{2i} = 0.5b_1 = 0.5(a_0 + c_0)$ and $b_{2i} = b_0$ ($i \geq 1$). Even if initially $a_0 \neq c_0$, a short period with extremely high temperatures will result in $a_i = c_i$.

In the practice of simulated annealing, temperature T drops from a very high to a very low value in many steps. In the *first phase*, $T \gg H(A) - H(B)$, while in the *third phase* $T \ll H(C) - H(B)$. In what I shall call the *second phase*, temperature T is, informally speaking, "in the magnitudes of" $H(A) - H(B)$ and $H(C) - H(B)$.

We have just seen that by the end of the first phase $a_I = c_I$. The precision of the algorithm will be determined by $a_{II} - c_{II}$ at the end of the second phase, since in the third phase the states get frozen. If a_{II}, b_{II} and c_{II} denote the population in each of the states after the second and before the third phase, then the precision of the algorithm is $\frac{a_{II}+0.5b_{II}}{N}$. Obviously, here the phases are idealized, and in reality their boundaries are not so clear. Yet, this idealization will contribute to our better understanding of simulated annealing.

Next, observe that from equations (14):

$$a_{i+1} - c_{i+1} = (a_i - c_i) + (c_i \cdot p_C(T_i) - a_i \cdot p_A(T_i)) \qquad (15)$$

At the beginning of the second phase $a_I = c_I$, and therefore:

$$a_{I+1} - c_{I+1} = (a_I - c_I) + a_I(p_C(T_I) - p_A(T_I)) \qquad (16)$$

Population a_i and c_i will begin to diverge if and only if there is a period during the simulation when $p_A(T_i) \neq p_C(T_i)$. Otherwise, $a_i - c_i$ remains constantly zero.

For instance, if $H(A) = H(C)$, then $p_A(T_i) = p_C(T_i)$ at all times. Therefore, independently of the original distributions, simulated annealing will return both states [A] and [C] in 50% of the cases.

Even if $H(A) > H(C)$, SA-OT offers no such period on Fig. 6 and tableau 3: due to reasons related to strict domination, the transition probabilities only depend on the fatal constraint (called the "highest uncanceled violation mark" by Prince & Smolensky, 1993/2004), which does not distinguish between $p_A(T_i)$ and $p_C(T_i)$. On the contrary, when standard simulated annealing is applied to symbolic harmonic grammar, eqs. (13)

ensure that $p_C(T) > p_A(T)$. So $a_i - c_i$ can turn positive at the beginning of the second phase. The higher $p_C(T_i) - p_A(T_i)$, the quicker the divergence between a_i and c_i. Moreover, the longer this period, the higher the precision of the algorithm. These are the two mechanisms that contribute to the success of simulated annealing, to its high precision.

By way of example, let us suppose that there is a period when $H(A) - H(B) \gg T \gg H(C) - H(B)$, that is when $p_A(T) \approx 0$ and $p_C(T) \approx 1$. Then, state [A] acts as a trap for the hedgehogs, while it is still possible to escape from [C]. A hedgehog will end up in state [C] only if whenever he is in [B], he decides to move to [C], and not to [A], which has a probability of 0.5 each time. Provided that this period of the algorithm lasts $2k$ iterations, our hedgehog ends up in [C] with a probability of 0.5^k, and in [A] with a probability of $1 - 0.5^k$. Consequently, the more iterations in this crucial phase of the simulation, the higher the precision of the algorithm. The precision converges to 100% as $2k$ grows.

This is the main idea behind simulated annealing, even if details are more complicated, even for this simple landscape. For instance, after a_i and c_i have diverged, $c_i \cdot p_C(T_i) - a_i \cdot p_A(T_i)$ does not need to stay positive in (15), and so $a_i - c_i$ will not necessarily grow forever. If simulated annealing is very ("infinitely") slow, the system may reach an equilibrium in which $a_i \cdot p_A(T_i) = c_i \cdot p_C(T_i)$. That will be the topic of the next subsection.

To summarize, in the *first* and *second* phases, the hedgehogs can escape local optima. In the *second* and *third* phases, the hedgehogs are attracted by the global optimum. The longer the *second* phase (that is, the more iterations are available in the second phase), the greater the chance that a random walker will end up in the global optimum.

5.3 Equilibrium of our system

Candidates [A], [B] and [C] have been called states, in which each of the N hedgehogs can be found. As opposed to these *states*, the whole system can be characterized with some *macrostate*: following the physical analogy, a macrostate of the whole system is a distribution (a_i, b_i, c_i) of the random walkers. Different random walkers can change their state, hence the *microstate* of the whole system can change (a third concept); yet, the macrostate does not alter as long as the overall distribution remains the same.

A macrostate is an *equilibrium state* if it does not change in time, that is, $a_{i+1} = a_i$, $b_{i+1} = b_i$ and $c_{i+1} = c_i$. Maybe no random walker moves: the system is frozen, thus the microstate is also invariable. But it can very much be the case that individual hedgehogs move from one state to another one, but the distribution remains the same. From eqs. (14) and (15) we conclude that the system is in equilibrium if and only if

$$\begin{aligned} a_i \cdot p_A(T) &= c_i \cdot p_C(T) \\ b_i &= a_i \cdot p_A(T) + c_i \cdot p_C(T) \end{aligned} \qquad (17)$$

If enough iterations are performed, then the system can converge to this state. Decreasing the temperature very ("infinitely") slowly allows the system to stay in this macrostate of equilibrium. Then

$$\frac{c_i}{a_i} = \frac{p_A}{p_C} = e^{\frac{H(B)-H(A)}{T} - \frac{H(B)-H(C)}{T}} = e^{\frac{H(C)-H(A)}{T}} \tag{18}$$

will steadily hold true. Gradually decreasing T to zero, results in $\frac{c_i}{a_i}$ also converging to zero. The larger the difference $H(A) - H(C)$, the faster this convergence. In sum, an "infinitely slow" annealing (parameter T decreased to zero in many-many steps) is characterized by a precision of 1: it will only return the global optimum [A], and never the other local optimum, [C].

As a side note, solving the equation system (14) in the fixed point – that is, for $a_{i+1} = a_i = a^*$, $b_{i+1} = b_i = b^*$ and $c_{i+1} = c_i = c^*$ – with $N = a^* + b^* + c^*$ leads us to

$$\begin{aligned} a^*(T) &= N\frac{p_c}{p_c + 2p_c p_c + p + a} \\ b^*(T) &= 2N\frac{p_a p_c}{p_c + 2p_c p_c + p + a} \\ c^*(T) &= N\frac{p_a}{p_c + 2p_c p_c + p + a} \end{aligned} \tag{19}$$

In turn, inserting the transition probabilities (13) into (19) yields a Boltzmann distribution (in fact, a Boltzmann distribution in which [B] corresponds to a degenerate state, that is, to two states that have the same harmony and have been collapsed into one) as the state of equilibrium:

$$\begin{aligned} a^*(T) &= \frac{N}{Z(T)}e^{\frac{H(A)}{T}} \\ b^*(T) &= 2\frac{N}{Z(T)}e^{\frac{H(B)}{T}} \\ c^*(T) &= \frac{N}{Z(T)}e^{\frac{H(C)}{T}} \end{aligned} \tag{20}$$

where N is the number of random walkers run in parallel, whereas

$$Z(T) = e^{\frac{H(A)}{T}} + 2 \cdot e^{\frac{H(B)}{T}} + e^{\frac{H(C)}{T}} \tag{21}$$

is called the *partition function.* As $T \to +0$, the largest term (the first one) will dominate the partition function, and therefore $\lim a^*(T) = N$, but $\lim b^*(T) = 0$ and $\lim c^*(T) = 0$.

Observe that *Maximum Entropy OT* (Goldwater & Johnson, 2003) postulates a distribution similar to (20) (ignoring the factor 2 in b^* and Z), as if annealing stopped at a positive value of temperature T.

The next section derives the main result of this paper. It shows that strict domination – postulated by Optimality Theory, and an asymptotic case in q-HG – allows cases in which $p_C(T) - p_A(T)$ is 1 in a crucial phase of the simulation (hence, simulated annealing is maximally efficient); but also cases in which $p_C(T) - p_A(T)$ is constantly 0, and therefore simulated annealing produces *irregular forms.*

6 Simulated annealing in the strict domination limit

6.1 Simulated annealing for q-HG

The present section contains the core message of this paper by asking what the consequences are of the $q \to +\infty$ *strict domination limit* for simulated annealing when it is applied to a q-HG grammar. We shall observe that strict domination can lead to very efficient computation, but also to severe errors.

Increasing q will increase the range of the objective function $H(w)$. It will also magnify the differences $H(x') - H(x)$ in the equation of the transition probability (11).

It follows that the *cooling schedule* must also be adapted. If the cooling schedule remained the same, even the highest temperatures would become very low compared to the differences in the objective function at high q values. In the strict domination limit, the algorithm would therefore miss its crucial first two phases, and start immediately with randomized gradient descent (Fig. 4), instead of simulated annealing (Fig. 5).

Therefore, the cooling schedule should be made a decreasing series of functions of q:

$$T_{max}[q] = T_0[q],\ T_1[q],\ T_2[q],\ \dots,\ T_i[q],\ \dots,\ T_L[q] = T_{min}[q] > 0 \tag{22}$$

Additionally, a cooling schedule will satisfy two requirements: first, we require that for some reasonable q_0 and for any $q > q_0$: $T_i[q] > T_{i+1}[q]$. Second, we also posit $\lim_{q\to+\infty} T_i[q] = +\infty$ for any i.

More specifically, the cooling schedule should be such that the three phases discussed in the previous section should be discernible. For any q, the first values in the series should be "much greater" than any possible difference in harmony of two neighboring candidates; which, based on (8), is in the order of magnitude of q^n. Similarly, the last values in the series should be for any q "much smaller" than the smallest possible difference in harmony, which is q, one violation difference of the lowest ranked constraint.

Bíró (2006) suggested using $T_i[q] = t_i \cdot q^{K_i}$, where K_i was decreased in an outer loop (from K_{max} to K_{min}, using K_{step}), and for each K_i, t_i was decreased from t_{max} to t_{min} by t_{step}. For instance, the case $K_{max} = 4$, $K_{min} = 0$, $K_{step} = 1$, $t_{max} = 3$, $t_{min} = 0.5$ and $t_{step} = 0.5$ would look like:

$$3 \cdot q^4,\ 2.5 \cdot q^4,\ 2 \cdot q^4, \dots, 0.5 \cdot q^4,\ 3 \cdot q^3, \dots, 0.5 \cdot q^3,\ 3 \cdot q^2, \dots, 0.5 \cdot q^0 \tag{23}$$

This kind of cooling schedule will be referred to as *linear*. Another option is to diminish the temperature exponentially (Biró, 2009):

$$T_i = (c \cdot q^n)^{\frac{m-i}{m}} \tag{24}$$

where n is the number of constraints in the q-HG grammar (the exponent of the highest ranked constraint). As one violation of the highest ranked constraint contributes q^n to the harmony function in (8), a large c (e.g., $c = 100$ – supposing that neighbors differ in only a few violations of constraint C_n) guarantees that initially the random walker will move freely: for any x and x', the transition probability $P(x \to x'|T_0) \approx 1$. At the same time, parameter m determines the speed of the cooling schedule: a large m diminishes the temperature only slowly. By the mth step, temperature is reduced to $T_m = 1$. The smallest possible difference in harmony – corresponding to a single violation of the lowest ranked constraint C_1 – is q. Therefore, if q is large, then $|H(x) - H(x')| \gg T_m$, which means that after m steps the system will have been frozen, the algorithm will have reached its third phase.

6.2 The V landscape: the good case

In order to understand the behavior of simulated annealing applied to q-Harmonic Grammar in the strict domination limit, let us return to the V landscape (Fig. 6). Recall equation (15), repeated here:

$$\begin{aligned} a_{i+1} - c_{i+1} &= (a_i - c_i) + (c_i \cdot p_C(T) - a_i \cdot p_A(T)) \\ &= (a_i - c_i)(1 - p_A(T)) + c_i(p_C(T) - p_A(T)) \end{aligned} \tag{25}$$

Remember that at the beginning of the second phase $a_i = c_i$. The speed at which the number of random walkers in states [A] and in [C] will diverge during the second phase therefore depends on $p_C(T) - p_A(T)$. If this value is close to zero, then the divergence will be very slow, and only an extremely large number of iterations can guarantee finding the globally optimal state with a high probability. If, however, there is a phase in the simulation (there is a value T) when $p_A(T)$ and $p_C(T)$ are very different, then the algorithm will be efficient.

The conclusion thus has been that the efficiency of simulated annealing depends crucially on the phase in which $p_A(T)$ is low and $p_C(T)$ is high. Is there such a phase in the strict domination limit? We shall see that in certain cases the strict domination limit makes simulated annealing extremely efficient, but not in other cases.

Let us start with the good case. Consider the following tableau for the V landscape (Fig. 6; $\alpha > \beta$, and both are positive integers):

	C$_\alpha$	C$_\beta$
[A]		*
[B]	*	*
[C]	*	

(26)

Suppose that all other constraints do not distinguish between the three candidates, and so they contribute the same constant term τ to the harmony. The harmony of the states are as follows: $H(A) = \tau - q^\beta$, $H(B) = \tau - q^\alpha - q^\beta$, and $H(C) = \tau - q^\alpha$. Consequently:

$$\begin{aligned} p_A(T) = P(A \to B|T) &= e^{-\frac{q^\alpha}{T}} \\ p_C(T) = P(C \to B|T) &= e^{-\frac{q^\beta}{T}} \end{aligned} \tag{27}$$

What we need is a cooling schedule with a second phase in which $p_C(T) - p_A(T)$ is large. A useful cooling schedule will start with $T_{max}[q] \gg q^\alpha$ and end with $T_{min}[q] \ll q^\beta$.

By having a sufficient number of intervening steps, there will be some $T_i[q] = q^\gamma$ where $\alpha > \gamma > \beta$. Such a cooling schedule can easily be constructed. In the case of a linear cooling schedule (23), use $t_{\max} = t_{\min} = 1$, $K_{\max} = \alpha + 0.5$ and $K_{\text{step}} = 1$. Alternatively, use $K_{\text{step}} < \alpha - \beta$, to make sure some $T_i[q] = t_i \cdot q^{K_i}$ falls between q^α and q^β. If you prefer the exponential cooling schedule scheme (24), then $m > n$ will make the exponent of q take some value between any two adjacent integers.

In turn, employing any of these cooling schedule schemes, let i be such that $T_i[q] = t_i \cdot q^\gamma$ with $\alpha > \gamma > \beta$. In this case,

$$\begin{aligned} \lim_{q\to+\infty} p_A(T_i[q]) &= \lim_{q\to+\infty} e^{-\frac{q^\alpha}{t_i \cdot q^\gamma}} = 0 \\ \lim_{q\to+\infty} p_C(T_i[q]) &= \lim_{q\to+\infty} e^{-\frac{q^\beta}{t_i \cdot q^\gamma}} = 1 \end{aligned} \tag{28}$$

Consequently, $p_C(T_i[q]) - p_A(T_i[q])$ converges to 1 in the strict domination limit. For large q, at iteration i, our random walking hedgehog is free to leave the locally optimal state [C], but is stuck in the global optimum [A].

This situation was already discussed in section 5.2, and we saw that the probability of ending up in [A] could be made to converge to 1 by increasing the number of steps in this phase of the algorithm. This can be achieved, for instance, by reducing the value of the t_{step} parameter in a linear cooling schedule (23), or by increasing m in an exponential cooling schedule (24). If $t_{step} < \frac{t_{max} - t_{min}}{2k}$, or if $m > \frac{2kn}{\alpha - \beta}$, then the algorithm will spend at least $2k$ iterations such that $q^\alpha > \mathcal{O}(T_i[q]) > q^\beta$, corresponding to a precision of at least $1 - 0.5^k$ in the strict domination limit.

Strict domination in this case has proven to be an asset. Increasing q also increases $p_C(T_i[q]) - p_A(T_i[q])$, and so simulated annealing is expected to work better.

6.3 The V landscape: the bad case

The situation will be very different with the following tableau (again, $\alpha > \beta$, and both are positive integers):

	C_α	C_β
[A]		
[B]	*	*
[C]		*

(29)

This time, $H(A) = \tau + 0$, $H(B) = \tau - q^\alpha - q^\beta$ and $H(C) = \tau - q^\beta$, whence

$$\begin{aligned} p_A(T) = P(A \to B|T) &= e^{-\frac{q^\alpha + q^\beta}{T}} \\ p_C(T) = P(C \to B|T) &= e^{-\frac{q^\alpha}{T}} \end{aligned} \tag{30}$$

What is $p_C(T) - p_A(T)$ in the strict domination limit?

$$\begin{aligned}\lim_{q\to+\infty}(p_C(T)-p_A(T)) &= \lim_{q\to+\infty} e^{-\frac{q^\alpha}{T}}\left(1-e^{-\frac{q^\beta}{T}}\right) = \\ &= \lim_{q\to+\infty} e^{-\frac{q^\alpha}{T}} \cdot \lim_{q\to+\infty}\left(1-e^{-\frac{q^\beta}{T}}\right) \qquad (31)\end{aligned}$$

This limit is always zero. Namely, if $T[q] < \mathcal{O}(q^\alpha)$, that is, if $\lim \frac{q^\alpha}{T[q]} = \infty$, then the first limit is zero and the second limit is less than or equal to 1. If, on the other hand, $T[q] > \mathcal{O}(q^\beta)$ (that is, $\lim \frac{q^\beta}{T[q]} = 0$), than the second limit is zero and the first limit is less than or equal to 1.

Thus, when simulated annealing is applied to a V landscape with tableau (29), the difference $p_C - p_A$ stays zero in the strict domination limit, at *any* temperature. The consequence of this fact for the dynamics in eq. (25) is that the probability of a hedgehog to be in state [A] or in state [C] will never diverge, yielding a 50% precision for *all* cooling schedules.

In summary, we have analyzed two variants of the asymmetric V landscape, displaying different behaviors in the strict domination limit. As the parameter q of a q-HG grammar is gradually increased, so does the behavior of simulated annealing approach the behavior of SA-OT. In the case of tableau (26), the precision converges to 100% as the number of iterations grows in the second phase; but it stays 50%, independently of the cooling schedule, for tableau (29). Next, we confirm this analysis with computer experiments.

7 Experiments with the V landscape

It is always good practice to also support the conclusions of an analytical discussion with computer experiments. Therefore, this section reports the results of simulations run in a V landscape with three states (candidates), as shown on Fig. 6. Can we confirm the above analyses of the grammars in tableaux (26) and (29)?

For the sake of concreteness, the two constraints were assigned weights q^2 and q respectively (i.e., $\alpha = 2$ and $\beta = 1$). Note that in both tableaux, the relative harmony of the three candidates are independent of q (*viz.*, $H_q([\mathrm{A}]) > H_q([\mathrm{C}]) > H_q([\mathrm{B}])$ for all q), not displaying any kind of cumulativity. Thus, the grammatical output is always [A].

The Java implementation of the simulated annealing algorithm on Fig. 5 was run on the *Atlasz HPC cluster* of the ELTE university. For each parameter combination discussed below, 10^6 random walks were launched, so that we could measure the *precision* of the algorithm by counting the frequency of returning the globally optimal candidate. With such a large sample size, the standard error of the population proportion (i.e., the precision) is below 10^{-3}. We also measured the distribution – mean and standard deviation – of the *length of the random* walk, that is, the number of iterations until convergence.

The three candidates were used by turns as the initial position of the random walk. The exponential cooling schedule followed (24), with variable i (initially 0) increased by 1 in each iteration. Parameters c and n were fixed: $c = 100$, to ensure a very high initial temperature, and $n = 2$, corresponding to the two constraints in the grammar. A step by the random walker increasing the violation of the higher ranked constraint decreases the harmony by q^2; when the temperature is initially $T_0 = c \cdot q^2$, even such a step has a probability of $\exp(-1/c) \approx 1$. After m steps, temperature dropped to $T_m = 1$; this point in time can be roughly seen as the beginning of the third phase, when temperature has become much lower than the smallest possible difference in harmony, q. Increasing parameter c increases how many of the first m iterations "kind-of" belong to the first phase, whereas decreasing parameter c increases the number of iterations in the "second phase". Remember that the success of the algorithm depends on the number of iterations in this second phase. In accordance with our prediction, simulations confirm that choosing a smaller c slightly improves the precision of the algorithm.

The stopping condition required the random walker not to move for $\ell = 60$ iterations. Recall that if the random walker is in position [B], it will always move to one of its neighbors. It will not move if it is in a local optimum, and the random number generated is higher than the transition probability (11). For this to happen 60 times, we must be extremely unlucky, unless the transition probability is already extremely low, as the consequence of a very low temperature. Reducing parameter ℓ to 40 or 20 will marginally decrease the precision of the algorithm and the average length of the random walk. Reducing it further to 10 will result in a more significant loss in precision, accompanied by an average length of the random walk diminished by a few steps. Indeed, given the very large sample size, it is not unexpected that a few times the random number generator will produce ten consecutive large values, stopping the algorithm prematurely.

For each of the two grammars analyzed above, the "good case" and the "bad case", we report in details the effect of tuning the two most interesting parameters: the base q, and m, the speed of simulated annealing. Tables 4 and 6 in the Appendix present the precision for each parameter combination. Given the sample size of 10^6 for this binary process (the output being either "correct" or "incorrect"), the standard error of the sample proportion, but also the error bar for the estimated proportion is below 0.1%.

Tables 5 and 7 present the speed of convergence: for each parameter combination, the mean and standard deviation of the 10^6 simulation lengths. Simulation length refers to the number of iterations until the random walker got stuck in the global or another local optimum: the value of the variable i in the cooling schedule (24) when the stopping condition becomes true minus ℓ, the number of iterations the random walker has been stuck here. Again due to the large sample size, the standard error of the mean in each cell is smaller by three orders of magnitude than the reported standard deviation.

The number of iterations is comparable to m, while it significantly decreases as q increases. An empirical law of the form $\textit{nr of iterations} = m^{\chi} \cdot (\ln q)^{\phi}$ approximates the observed data reasonably well, even though a closer look at the tables reveals a more complex behavior. Fitting the output of a separate set of experiments, we obtained $\chi = 0.954$ and $\phi = -0.351$ for the "good case", and $\chi = 0.945$ and $\phi = -0.361$ for the "bad case" (the difference between the two cases is highly significant). The conclusion is clear: simulated annealing with a given cooling schedule – in our case, a specific value of the parameters c and m – becomes faster in the strict domination limit.

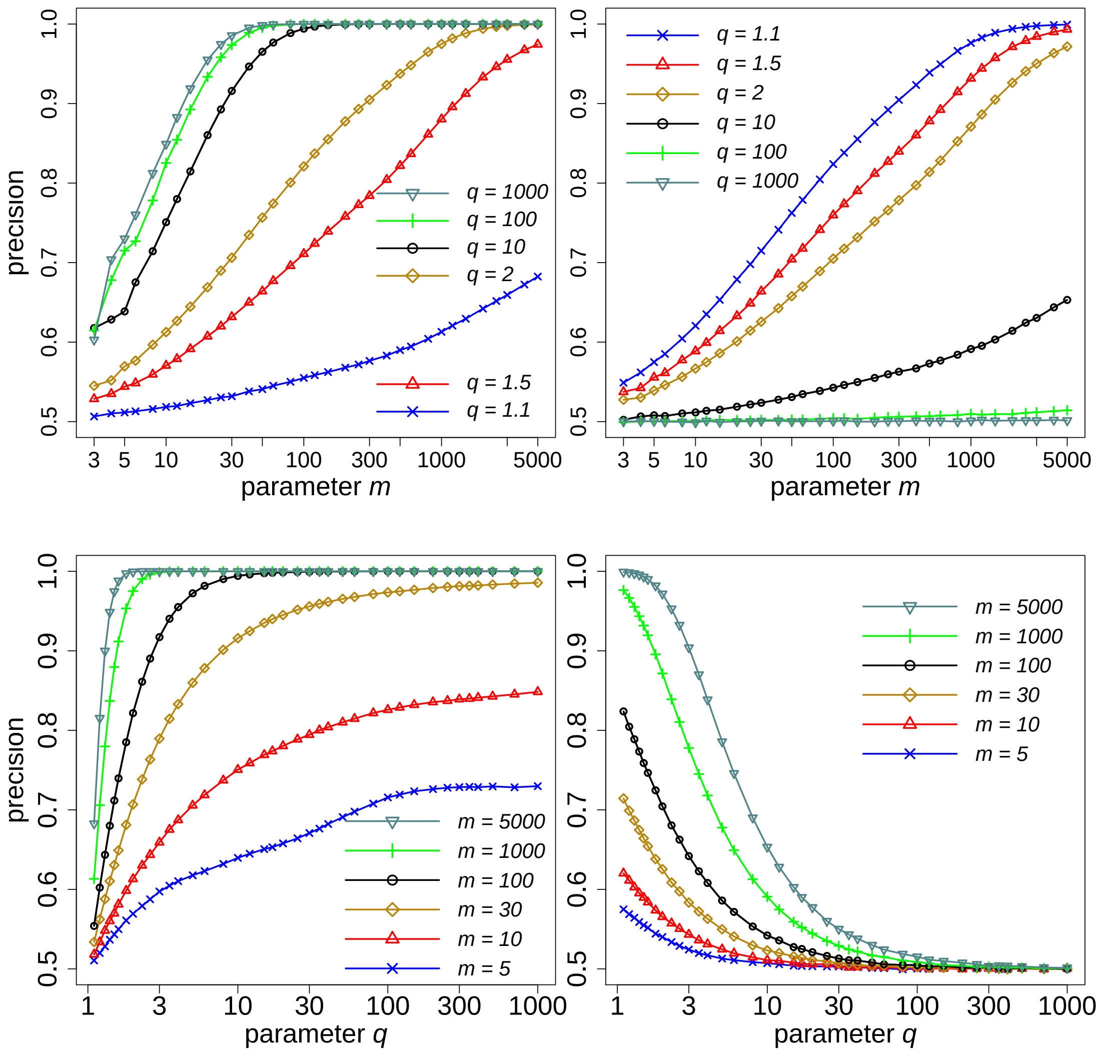

Figure 7: Precision of simulated annealing q-HG, as a function of parameters m and q, for the two grammars discussed: (26) shown on the left panels, and (29) on the right panels. For a given q, increasing m always improves precision. However, as q grows from 1.1 to 1000 (on a logarithmic scale), the two grammars display opposite behaviors.

Returning to precision, the plots on Fig. 7 present the results of yet another set of simulations, each data point measured with 10^6 runs. Similarly to Tables 4 and 6, we can observe how for a given q, increasing the number of iterations (increasing m) improves precision. This behavior is not at all surprising, as a q-HG grammar is a real-valued optimization problem. From the general convergence properties of "standard" simulated annealing, we know that a sufficiently slow cooling schedule will produce high precision. Mathematically speaking, I conjecture that for any q and all $\epsilon > 0$ there exists an m such that a specific q-HG grammar will yield a precision higher than $1 - \epsilon$.

Yet, this statement with a reversed scope is not necessarily true. Observe the plots as q grows. In the case of the "good grammar" on the left panels, the strict domination limit corresponds to some precision between 50% and 100%, depending on m. It reminds us of the precision of SA-OT with the same tableau, which also depends on the cooling schedule. A large q paired with a large m easily yields a precision sufficiently close to 1. But such is not the case on the right panels, corresponding to the "bad grammar".

The formal analysis in the previous section and the current experimental results both suggest that no cooling schedule is good enough for all q-HG grammars based on the "bad case" tableau (29). In fact it seems that for any cooling schedule – probably even beyond the exponential cooling schedule scheme (24) – a sufficiently large q will yield a precision close enough to the exactly 50% precision observed for the SA-OT Algorithm:

Conjecture 3. *For any cooling schedule and for all* $\epsilon > 0$*, there exists a* $q_0 > 1$ *such that for all* $q > q_0$ *the precision of a q-HG grammar with* (29) *is less than* $0.5 + \epsilon$*.*

8 Summary: why is it human to err?

From the old joke with the three greengrocers we learn that optimizing locally is more convenient for the human brain than optimizing globally. But then, the hedgehog in the optimization procedure may climb the wrong hill, producing an error. Therefore, we conclude that to err is human.

We have compared simulated annealing with a real-valued harmony function, as it happens in connectionist and symbolic harmonic grammars, to simulated annealing with strict domination. In the former case, one can choose a sufficiently slow cooling schedule so that the precision of the algorithm (the probability of returning the global optimum) be greater than $1-\epsilon$: the precision can be made to converge to 100%. This is not the case with strict domination, however. With some grammars – i.e., constraint hierarchies, and candidate sets with neighborhood structures – the precision of the *Simulated Annealing for Optimality Theory Algorithm* (SA-OT) does not converge to 1. The same applies to q-HG: if q is large enough, the precision can be far away from 100%.

Encouraged by Newmeyer (1983) and slightly diverging from standard terminology, I suggest using the phrase *grammatical form* for linguistic forms predicted by a grammar, such as the global optimum in OT-style frameworks (1). A grammar is a model of the native speaker's linguistic competence, "the speaker-hearer's knowledge of [their] language" (Chomsky, 1965); whereas the implementation of this grammar should mirror the speaker's linguistic performance (Smolensky & Legendre, 2006; Bíró, 2006).

Jackendoff (2007, p. 27) explains Chomsky's 'knowledge' as "whatever is in speaker's heads that enables them to speak and understand their native language(s)". But what is in one's head? A network of neurons. Hence, the motivation to bridge connectionist harmonic grammars to symbolic ones, and then to OT grammars, via q-HG. Now, should a grammar be an adequate description of the speaker's knowledge, the grammar will correctly predict the forms produced and judged as acceptable – provided a perfect implementation thereof.

In an imperfect implementation, however, errors occur: forms that are not grammatical, but are nevertheless produced. These could be called *performance errors*. Yet, this term has been employed differently, and so let me suggest two alternatives. Some of the erroneous forms occur more frequently if the production algorithm is run more quickly: these could be seen as *fast speech forms* in a broad sense. Whereas other forms emerge independently of the production speed, at least in OT and in the strict domination limit of q-HG: these can be identified as *irregular forms*. An example is progressive voice assimilation in some special cases in Dutch, a language which otherwise displays regressive voice assimilation exclusively, "as a rule" (Bíró, 2006).

The moral is that linguists need not struggle to have their grammars encompass each and every form accepted by the native speaker. It might be a more fruitful strategy to discount some forms, and to aim at a simpler grammar. Then, the irregular forms contravening the general "rules" may simply turn out to be errors made by the grammars that do not count.

References

Bíró, T. (2003). Quadratic alignment constraints and finite state Optimality Theory. In *Proceedings of the workshop on Finite-State Methods in Natural Language Processing (FSMNLP), held within EACL-03, Budapest* (pp. 119–126). (Also: ROA-600[1])

Bíró, T. (2005a). When the hothead speaks: Simulated Annealing Optimality Theory for Dutch fast speech. In C. Cremers, H. Reckman, M. Poss, & T. van der Wouden (Eds.), *Proc. of 15th Computational Linguistics in the Netherlands Conf.* Leiden.

Bíró, T. (2005b). How to define Simulated Annealing for Optimality Theory? In *Proceedings of Formal Grammar 10 and Mathematics of Language 9.* Edinburgh.

Bíró, T. (2006). *Finding the right words: Implementing Optimality Theory with simulated annealing.* Phd thesis, University of Groningen. (ROA-896)

Biró, T. (2007). The benefits of errors: Learning an OT grammar with a structured candidate set. In *Proceedings of the workshop on Cognitive Aspects of Computational Language Acquisition* (pp. 81–88). Prague: Assoc. for Computational Linguistics.

Biró, T. (2009). Elephants and optimality again: SA-OT accounts for pronoun resolution in child language. In B. Plank, E. Tjong Kim Sang, & T. Van de Cruys (Eds.), *Computational Linguistics in the Netherlands 2009* (pp. 9–24). Groningen: LOT.

Biró, T. (2013). Towards a robuster interpretive parsing: Learning from overt forms in Optimality Theory. *Journal of Logic, Language and Information*, *22*(2), 139–172.

[1]ROA stands for *Rutgers Optimality Archive* at `http://roa.rutgers.edu/`.

Boersma, P. (1997). How we learn variation, optionality, and probability. *Proceedings of the Institute of Phonetic Sciences, Amsterdam (IFA)*, *21*, 43–58.

Boersma, P. (2011). A programme for bidirectional phonology and phonetics and their acquisition and evolution. In A. Benz & J. Mattausch (Eds.), *Bidirectional Optimality Theory* (pp. 33–72). Amsterdam: John Benjamins.

Boersma, P., & Hayes, B. (2001). Empirical tests of the Gradual Learning Algorithm. *Linguistic Inquiry*, *32*, 45–86. (Also: ROA-348)

Boersma, P., & Pater, J. (2016 [2008]). Convergence properties of a gradual learning algorithm for Harmonic Grammar. In J. J. McCarthy & J. Pater (Eds.), *Harmonic Grammar and Harmonic Serialism.* Equinox. (First published on ROA in 2008.)

Černy, V. (1985). Thermodynamical approach to the travelling salesman problem. *Journal of Optimization Theory and Applications*, *45*, 41–51.

Chomsky, N. (1965). *Aspects of the theory of syntax.* Cambridge, Mass.: MIT Press.

Eisner, J. (1997). Efficient generation in Primitive Optimality Theory. In *Proceedings of the 35th annual meeting of the Association for Computational Linguistics (ACL-1997) and 8th EACL* (pp. 313–320). Madrid. (Also: ROA-206)

Eisner, J. (2000). Easy and hard constraint ranking in Optimality Theory: Algorithms and complexity. In J. Eisner, L. Karttunen, & A. Thériault (Eds.), *Finite-state phonology: Proc. of the 5th SIGPHON workshop* (pp. 22–33). Luxembourg.

Gigerenzer, G., Todd, P. M., & the ABC Research Group. (1999). *Simple heuristics that make us smart.* Oxford: Oxford University Press.

Goldwater, S., & Johnson, M. (2003). Learning OT constraint rankings using a maximum entropy model. In J. Spenader, A. Eriksson, & O. Dahl (Eds.), *Proceedings of the Stockholm workshop on Variation within Optimality Theory* (pp. 111–120).

González, C. (2005). Segmental alternations in Yaminahua. In *Proc. 24th West Coast Conference in Formal Linguistics, Simon Fraser University, Vancouver, Canada.*

Graf, T. (2017). The power of locality domains in phonology. *Phonology*, *34*, 1–21.

Heinz, J., Kobele, G. M., & Riggle, J. (2009). Evaluating the complexity of Optimality Theory. *Linguistic Inquiry*, *40*(2), 277–288.

Henderson, D., Jacobson, S. H., & Johnson, A. W. (2003). The theory and practice of simulated annealing. In F. Glover & G. A. Kochenberger (Eds.), *Handbook of metaheuristics* (Vol. 57, pp. 287–319). New York, etc.: Kluwer.

Idsardi, W. J. (2006a). A simple proof that Optimality Theory is computationally intractable. *Linguistic Inquiry*, *37*, 271–275.

Idsardi, W. J. (2006b). *Misplaced optimism.* ROA-840.

Jackendoff, R. (2007). *Language, consciousness, culture: Essays on mental structure.* Cambridge, Mass.: MIT Press.

Jäger, G. (2002). Some notes on the formal properties of bidirectional Optimality Theory. *Journal of Logic, Language and Information*, *11*(4), 427–451.

Jäger, G., & Rosenbach, A. (2006). The winner takes it all – almost: Cumulativity in grammatical variation. *Linguistics*, *44*(5), 937–971.

Karttunen, L. (2006). The insufficiency of paper-and-pencil linguistics: the case of Finnish prosody. In R. M. Kaplan, M. Butt, M. Dalrymple, & T. H. King (Eds.), *Intelligent linguistic architectures* (pp. 287–300). Stanford: CSLI.

Kirkpatrick, S., Jr., C. D. G., & Vecchi, M. P. (1983). Optimization by simulated annealing. *Science*, *220*(4598), 671–680.

Kornai, A. (2006a). *Is OT NP-hard?* ROA-838.

Kornai, A. (2006b). *Guarded optimalism.* ROA-841.

Kornai, A. (2008). *Mathematical linguistics.* London: Springer-Verlag.

Magri, G. (2016). Error-driven learning in Optimality Theory and Harmonic Grammar: A comparison. *Phonology*, *33*(3), 493–532.

McCarthy, J. J. (2002). *Against gradience.* ROA-510.

McCarthy, J. J. (2003). OT constraints are categorical. *Phonology*, *20*(1), 75–138.

McCarthy, J. J. (2007). Restraint of analysis. In S. Blaho, P. Bye, & M. Krämer (Eds.), *Freedom of analysis* (pp. 203–231). Berlin – New York: Mouton de Gruyter.

McCarthy, J. J. (2010). An introduction to Harmonic Serialism. *Language and Linguistics Compass*, *4*(10), 1001–1018.

McCarthy, J. J., & Prince, A. (1995). Faithfulness and reduplicative identity. In J. Beckman, S. Urbanczyk, & L. W. Dickey (Eds.), *Papers in Optimality Theory* (pp. 249–384). Amherst: Graduate Linguistic Students' Association, U. Mass.

Metropolis, N., Rosenbluth, A. W., Rosenbluth, M. N., Teller, A. H., & Teller, E. (1953). Equation of state calculation by fast computing machines. *Journal of Chemical Physics*, *21*(6), 1087–1092.

Newmeyer, F. J. (1983). *Grammatical theory: Its limits and its possibilities.* Chicago: University of Chicago Press.

Pater, J. (2009). Weighted constraints in generative linguistics. *Cognitive Science*, *33*(6), 999–1035.

Prince, A. (2002). *Anything goes.* ROA-536.

Prince, A., & Smolensky, P. (1993/2004). *Optimality Theory: Constraint interaction in Generative Grammar.* Malden, MA, etc.: Blackwell. (Originally as *Technical Report nr. 2. of the Rutgers University Center for Cognitive Science*, 1993.)

Pulleyblank, D., & Turkel, W. J. (2000). Learning phonology: Genetic algorithms and Yoruba tongue-root harmony. In J. Dekkers, F. van der Leeuw, & J. van De Weijer (Eds.), *Optimality Theory: Phonology, syntax, and acquisition* (pp. 554–591). Oxford: Oxford University Press.

Reeves, C. R. (Ed.). (1995). *Modern heuristic techniques for combinatorial problems.* London, etc.: McGraw-Hill.

Smolensky, P., & Legendre, G. (Eds.). (2006). *The Harmonic Mind: From neural computation to Optimality-Theoretic grammar.* Cambridge, MA – London, UK: MIT Press.

Soderstrom, M., Mathis, D. W., & Smolensky, P. (2006). Abstract genomic encoding of universal grammar in Optimality Theory. In P. Smolensky & G. Legendre (Eds.), *The Harmonic Mind* (Vol. 2, pp. 403–471). Cambridge – London: MIT Press.

Tesar, B., & Smolensky, P. (2000). *Learnability in Optimality Theory.* Cambridge, MA – London, UK: The MIT Press.

Turkel, B. (1994). *The acquisition of Optimality Theoretic systems.* m.s., ROA-11.

Watanabe, S. (2009). *Cultural and educational contributions to recent phonological changes in Japanese.* Phd thesis, The University of Arizona.

Appendix: Numerical results of the computer experiments

$m =$	10	20	50	100	200	500	1000	2000	5000
$q =$1.1	*0.518*	*0.527*	*0.541*	*0.555*	*0.568*	*0.589*	*0.612*	*0.642*	*0.682*
1.2	*0.534*	*0.550*	*0.578*	*0.601*	*0.627*	*0.665*	*0.705*	*0.753*	*0.815*
1.5	*0.572*	*0.607*	*0.665*	*0.711*	*0.758*	*0.822*	*0.881*	*0.933*	*0.975*
2.0	*0.612*	*0.669*	*0.757*	*0.821*	*0.877*	*0.937*	*0.975*	*0.994*	*0.999*
3.0	*0.659*	*0.739*	*0.850*	*0.918*	*0.962*	*0.991*	*0.999*	*1.000*	*1.000*
5.0	*0.705*	*0.803*	*0.920*	*0.972*	*0.994*	*1.000*	*1.000*	*1.000*	*1.000*
10	*0.751*	*0.860*	*0.965*	*0.994*	*1.000*	*1.000*	*1.000*	*1.000*	*1.000*
20	*0.780*	*0.894*	*0.983*	*0.999*	*1.000*	*1.000*	*1.000*	*1.000*	*1.000*
50	*0.810*	*0.922*	*0.993*	*1.000*	*1.000*	*1.000*	*1.000*	*1.000*	*1.000*
100	*0.826*	*0.934*	*0.995*	*1.000*	*1.000*	*1.000*	*1.000*	*1.000*	*1.000*
200	*0.836*	*0.942*	*0.997*	*1.000*	*1.000*	*1.000*	*1.000*	*1.000*	*1.000*
500	*0.842*	*0.950*	*0.998*	*1.000*	*1.000*	*1.000*	*1.000*	*1.000*	*1.000*
1000	*0.849*	*0.955*	*0.998*	*1.000*	*1.000*	*1.000*	*1.000*	*1.000*	*1.000*

Table 4: Precision of the good case grammar (26).

$m =$	10	20	50	100	200	500	1000	2000	5000
$q =$1.1	*10.98* ±*1.59*	*22.01* ±*2.43*	*56.65* ±*4.59*	*116.55* ±*7.70*	*239.70* ±*13.22*	*614.41* ±*26.14*	*1232.45* ±*43.56*	*2450.88* ±*73.90*	*6046.75* ±*152.49*
1.2	*10.69* ±*1.56*	*21.38* ±*2.39*	*54.96* ±*4.52*	*113.01* ±*7.60*	*232.27* ±*13.13*	*595.35* ±*26.27*	*1194.37* ±*43.96*	*2374.62* ±*74.55*	*5854.86* ±*153.41*
1.5	*10.02* ±*1.51*	*19.95* ±*2.33*	*51.01* ±*4.46*	*104.57* ±*7.58*	*214.35* ±*13.26*	*548.75* ±*27.16*	*1099.99* ±*45.28*	*2184.72* ±*75.08*	*5379.69* ±*148.67*
2.0	*9.31* ±*1.49*	*18.37* ±*2.32*	*46.58* ±*4.50*	*94.94* ±*7.70*	*193.73* ±*13.32*	*494.83* ±*26.78*	*991.84* ±*42.77*	*1970.70* ±*67.90*	*4857.47* ±*132.46*
3.0	*8.50* ±*1.49*	*16.56* ±*2.36*	*41.36* ±*4.58*	*83.56* ±*7.60*	*169.50* ±*12.45*	*432.51* ±*23.34*	*869.48* ±*36.09*	*1732.28* ±*58.21*	*4277.83* ±*116.03*
5.0	*7.71* ±*1.51*	*14.76* ±*2.42*	*36.13* ±*4.53*	*72.24* ±*7.00*	*145.99* ±*10.68*	*373.31* ±*19.48*	*753.75* ±*31.07*	*1505.55* ±*51.08*	*3723.11* ±*102.70*
10	*6.92* ±*1.56*	*12.92* ±*2.48*	*30.77* ±*4.31*	*60.94* ±*6.09*	*122.96* ±*8.87*	*315.16* ±*16.69*	*638.67* ±*27.16*	*1279.13* ±*44.74*	*3168.20* ±*89.88*
20	*6.35* ±*1.61*	*11.55* ±*2.52*	*26.82* ±*4.06*	*52.77* ±*5.41*	*106.34* ±*7.86*	*272.64* ±*14.86*	*553.96* ±*24.46*	*1111.90* ±*40.24*	*2758.20* ±*80.46*
50	*5.76* ±*1.65*	*10.18* ±*2.53*	*23.00* ±*3.78*	*44.91* ±*4.86*	*90.26* ±*7.00*	*231.19* ±*13.14*	*470.90* ±*21.77*	*947.91* ±*35.78*	*2355.86* ±*70.99*
100	*5.43* ±*1.67*	*9.40* ±*2.54*	*20.83* ±*3.65*	*40.42* ±*4.58*	*81.00* ±*6.51*	*207.30* ±*12.12*	*422.72* ±*20.16*	*852.58* ±*33.13*	*2122.12* ±*65.46*
200	*5.18* ±*1.69*	*8.77* ±*2.55*	*19.07* ±*3.54*	*36.77* ±*4.39*	*73.50* ±*6.14*	*187.85* ±*11.30*	*383.30* ±*18.83*	*774.45* ±*30.92*	*1930.45* ±*60.88*
500	*4.91* ±*1.72*	*8.09* ±*2.55*	*17.21* ±*3.44*	*32.89* ±*4.17*	*65.49* ±*5.75*	*167.04* ±*10.38*	*341.04* ±*17.31*	*690.54* ±*28.44*	*1724.52* ±*55.81*
1000	*4.73* ±*1.75*	*7.68* ±*2.55*	*16.05* ±*3.38*	*30.50* ±*4.06*	*60.52* ±*5.50*	*154.14* ±*9.81*	*314.71* ±*16.35*	*638.14* ±*26.91*	*1595.72* ±*52.61*

Table 5: Number of iterations in the good case grammar (26).

$m =$	10	20	50	100	200	500	1000	2000	5000
$q =$1.1	*0.621*	*0.679*	*0.762*	*0.824*	*0.877*	*0.938*	*0.976*	*0.994*	*0.999*
1.2	*0.611*	*0.666*	*0.745*	*0.806*	*0.858*	*0.922*	*0.966*	*0.990*	*0.999*
1.5	*0.589*	*0.634*	*0.704*	*0.760*	*0.811*	*0.878*	*0.931*	*0.971*	*0.993*
2.0	*0.567*	*0.601*	*0.658*	*0.706*	*0.752*	*0.814*	*0.871*	*0.927*	*0.972*
3.0	*0.544*	*0.568*	*0.608*	*0.642*	*0.678*	*0.728*	*0.778*	*0.836*	*0.903*
5.0	*0.526*	*0.539*	*0.565*	*0.587*	*0.611*	*0.644*	*0.678*	*0.722*	*0.785*
10	*0.513*	*0.518*	*0.532*	*0.542*	*0.555*	*0.574*	*0.591*	*0.614*	*0.653*
20	*0.506*	*0.509*	*0.514*	*0.520*	*0.527*	*0.536*	*0.545*	*0.557*	*0.576*
50	*0.502*	*0.504*	*0.506*	*0.508*	*0.510*	*0.513*	*0.517*	*0.522*	*0.529*
100	*0.500*	*0.502*	*0.502*	*0.504*	*0.505*	*0.507*	*0.508*	*0.511*	*0.514*
200	*0.500*	*0.500*	*0.501*	*0.502*	*0.502*	*0.503*	*0.504*	*0.506*	*0.508*
500	*0.501*	*0.500*	*0.500*	*0.502*	*0.501*	*0.501*	*0.502*	*0.502*	*0.503*
1000	*0.500*	*0.500*	*0.500*	*0.501*	*0.501*	*0.500*	*0.500*	*0.501*	*0.502*

Table 6: Precision of the bad case grammar (29).

$m =$	10	20	50	100	200	500	1000	2000	5000
$q =$1.1	*10.14* ±*1.69*	*20.22* ±*2.68*	*51.71* ±*5.26*	*105.81* ±*9.03*	*216.63* ±*15.74*	*552.39* ±*31.23*	*1103.37* ±*49.89*	*2186.02* ±*80.05*	*5373.92* ±*157.94*
1.2	*9.85* ±*1.64*	*19.61* ±*2.59*	*50.16* ±*5.06*	*102.73* ±*8.70*	*210.45* ±*15.16*	*537.24* ±*30.37*	*1073.91* ±*48.89*	*2128.33* ±*78.65*	*5232.02* ±*154.65*
1.5	*9.16* ±*1.54*	*18.21* ±*2.39*	*46.57* ±*4.59*	*95.50* ±*7.86*	*195.92* ±*13.76*	*501.70* ±*28.10*	*1005.05* ±*46.17*	*1993.59* ±*75.06*	*4901.74* ±*147.26*
2.0	*8.41* ±*1.43*	*16.66* ±*2.19*	*42.58* ±*4.13*	*87.41* ±*7.01*	*179.55* ±*12.19*	*461.50* ±*25.14*	*927.14* ±*42.06*	*1842.21* ±*69.73*	*4533.82* ±*137.79*
3.0	*7.56* ±*1.32*	*14.86* ±*1.98*	*37.90* ±*3.67*	*77.84* ±*6.12*	*160.10* ±*10.56*	*413.03* ±*21.79*	*833.04* ±*36.52*	*1659.85* ±61.44	*4092.47* ±*123.89*
5.0	*6.70* ±*1.21*	*13.07* ±*1.80*	*33.20* ±*3.27*	*68.14* ±*5.39*	*140.28* ±*9.17*	*362.96* ±*18.78*	*734.92* ±*31.33*	*1468.62* ±*52.62*	*3629.50* ±*107.00*
10	*5.83* ±*1.11*	*11.22* ±*1.62*	*28.32* ±*2.90*	*58.05* ±*4.73*	*119.51* ±*7.94*	*309.87* ±*16.23*	*629.85* ±*26.94*	*1262.57* ±*44.85*	*3128.02* ±*90.77*
20	*5.17* ±*1.06*	*9.83* ±*1.50*	*24.63* ±*2.63*	*50.42* ±*4.26*	*103.75* ±*7.10*	*269.27* ±*14.46*	*548.93* ±*24.16*	*1103.20* ±*40.01*	*2738.12* ±*80.23*
50	*4.50* ±*0.96*	*8.45* ±*1.37*	*20.98* ±*2.37*	*42.85* ±*3.80*	*88.11* ±*6.29*	*228.78* ±*12.75*	*467.81* ±*21.47*	*943.04* ±*35.44*	*2345.53* ±*70.64*
100	*4.12* ±*0.88*	*7.65* ±*1.29*	*18.86* ±*2.22*	*38.43* ±*3.52*	*78.99* ±*5.81*	*205.15* ±*11.74*	*420.16* ±*19.87*	*848.85* ±*32.77*	*2114.38* ±*65.13*
200	*3.84* ±*0.84*	*7.00* ±*1.23*	*17.13* ±*2.09*	*34.83* ±*3.30*	*71.53* ±*5.41*	*185.80* ±*10.88*	*381.02* ±*18.51*	*771.42* ±*30.52*	*1924.41* ±*60.45*
500	*3.54* ±*0.86*	*6.30* ±*1.15*	*15.28* ±*1.94*	*30.98* ±*3.05*	*63.56* ±*4.98*	*165.08* ±*9.95*	*338.97* ±*17.01*	*687.90* ±*28.17*	*1719.52* ±*55.43*
1000	*3.33* ±*0.86*	*5.88* ±*1.11*	*14.13* ±*1.85*	*28.58* ±*2.89*	*58.59* ±*4.71*	*152.17* ±*9.38*	*312.68* ±*16.07*	*635.62* ±*26.62*	*1591.19* ±*52.31*

Table 7: Number of iterations in the bad case grammar (29).