Towards Fast Cognate Alignment on Imbalanced Data

Logan Born[1], **M. Willis Monroe**[2], **Kathryn Kelley**[3], **Anoop Sarkar**[1]
[1]Simon Fraser University, [2]University of New Brunswick, [3]Università di Bologna
loborn@sfu.ca, willis.monroe@unb.ca, kathrynerin.kelley@unibo.it, anoop@cs.sfu.ca

Abstract

Cognate alignment models purport to enable decipherment, but their speed and need for clean data can make them unsuitable for realistic decipherment problems. We seek to draw attention to these shortcomings in the hopes that future work may avoid them, and we outline two techniques which begin to overcome the described problems.

Keywords: cognate detection, alignment, decipherment

1. Introduction

Cognate alignment models aim to identify cognate lexeme pairs by aligning word lists from related languages; when one of these lists comes from an undeciphered language, this alignment produces a decipherment for any words which are correctly mapped to their cognates. In this study, we highlight limitations of current cognate alignment models which restrict their usefulness in practical decipherment tasks, and propose partial solutions for more realistic data.

2. Motivation

The state-of-the art for cognate alignment (Tamburini, 2023) uses coupled simulated annealing (Xavier-de-Souza et al., 2010) to solve a search problem over the set of all k-permutations mapping n lost-language words into k known-language buckets. The prior state-of-the-art Luo et al. 2019 alternated between learning character- and word-level pairings using iterative expectation-maximization-style training. Both achieve strong results when every word has at least one cognate in the language it has been paired with; however, this is artificially clean compared to true decipherment, where not only are unpaired words likely, but there may also be uncertainties about the underlying word boundaries or character inventory. Luo et al. (2019) acknowledge this and evaluate on less-clean Ugaritic-Old Hebrew data, in which setting their accuracy drops to just 65.9%, from 93.5% on clean data. Tamburini 2023 does not include noisy evaluations, likely because their search is computationally intensive and does not appear scalable to settings with many unpaired words.

Luo et al. (2019, 3152) also "found it beneficial to train [...] only on a randomly selected subset (10%) of the entire corpus with the same percentage of noncognates," but observe that such filtering is impossible in a realistic setting where it is not known which words *have* cognates. On undeciphered data, both word lists would need to be sampled independently, meaning that a 10% subset of the corpus should be expected to contain a mere 1% of the original cognate pairs, destroying most of the signal that the model would learn from.

Finally, note that existing models can be inefficient even on clean data (Tamburini 2023, 89 "took a relevant time to converge"), which is undesirable for real decipherments where the correct target language is not known beforehand. Such settings may require aligning to multiple targets before a true solution can be found, and this scattershot approach is only feasible when each individual language pair can be aligned efficiently.

3. LMs for Character-Level Alignment

Tran (2020) considers the task of adapting pretrained models to new languages. Given a matrix of pretrained word embeddings $\mathbf{E}_e$, they propose to derive embeddings $\mathbf{E}_f$ for a new language using linear combinations of the rows of $\mathbf{E}_e$:

$$\mathbf{E}_f[i] = \sum_{j=1}^{|V_e|} \alpha_{ij}\mathbf{E}_e[j] = \alpha_i \mathbf{E}_e \qquad (1)$$

where α_i is a sparse weight vector satisfying $\sum_j^{|V_e|} \alpha_{ij} = 1$. The authors cite two offline approaches (Dyer et al., 2013; Conneau et al., 2017) which can be used to estimate α by exploiting sentence-level context. If the dependence on sentence-level context (which is absent from the word lists used for cognate alignment) could be eliminated, we suggest that a similar technique could by applied to the embeddings from a *character*-level language model to quickly learn character equivalencies between two scripts or languages as a first step towards cognate alignment.

Concretely, we propose to first train a character-level language model on the known language. The inputs to the model are individual words from the known-language word list, with no additional context, tokenized at the character level. Given the

Second Workshop on Computation and Written Language (CAWL 2024) @LREC-COLING-2024, pages 53–58

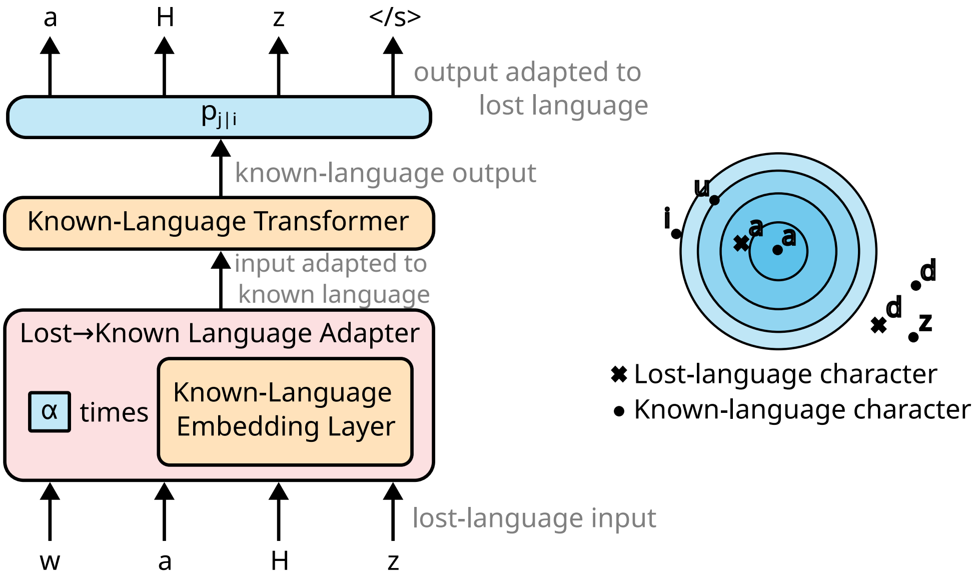

Figure 1: Schematic view of proposed architecture (left) and sampling procedure for Equation 2 (right). Characters are embedded using a matrix E (for the known language) or αE (for the lost language). The probability that character a in the known-language corresponds to character a in the lost language is proportional to their distance in this embedding space. During fine-tuning, α (and by extension $p_{j|i}$, which depends on α) is the only parameter that is allowed to be updated.

small size of cognate detection datasets, we use a shallow Transformer (Vaswani et al., 2017) with a small feature dimension (2 layers, 4 heads, and 32 dimensions); this is the largest model which we are able to train reliably, as deeper models or those with larger dimension often fail to converge. We use positional encodings (Vaswani et al., 2017) and apply dropout at a rate of 0.5. This model is trained with SGD to minimize categorical cross-entropy on an autoregressive language modeling task using a causal attention mask.

Let $E \in \mathbb{R}^{k\times 32}$ be the embedding layer of this model, where k is the number of known-language characters. Let $M : \mathbb{R}^{n\times 32} \rightarrow \mathbb{R}^{n\times k}$ be a black-box representation for the remainder of the model, which maps a sequence of n 32-dimensional character embeddings onto a sequence of log-probabilities over k known characters.

We next introduce a mapping $\alpha \in \mathbb{R}^{l\times k}$ following Tran 2020. The product $\alpha E \in \mathbb{R}^{l\times 32}$ can be seen as an embedding matrix for l distinct lost-language characters, and $M \circ \alpha E$ can be seen as a hybrid language model which accepts lost-language inputs and predicts known-language outputs. To convert the outputs from $M \circ \alpha E$ into a distribution over lost-language characters, we introduce the following conditional probability distribution inspired by t-SNE (Hinton and Roweis, 2002; van der Maaten and Hinton, 2008):

$$\log p_{j|i} = \frac{-|\mathbf{x}_i - \alpha E_j|^2/2\sigma_i}{\sum_{h\neq i} -|\mathbf{x}_i - \alpha E_h|^2/2\sigma_i} \tag{2}$$

where $1 \leq i \leq k$, $1 \leq j \leq l$, $\mathbf{x}_i$ is the embedding for the ith known character, αE_j is the embedding for the jth lost character, and σ_i is a per-character density estimate. Given a known-language character i, suppose we sample neighboring characters in the embedding space based on their distance from $\mathbf{x}_i$, with Gaussian falloff. Assuming we are only allowed to sample neighbors from the lost language, $p_{j|i}$ models the probability that the lost-language character j will be the one sampled. This is equivalent to the sampling procedure used in t-SNE (van der Maaten and Hinton, 2008) with the modification that points are divided into two classes (known and lost), and each class can only sample points from the other class.

Given a distribution $\mathbf{p} = [p_1, ..., p_k]$ over known-language characters returned as output by $M \circ \alpha E$, we model the corresponding distribution $\tilde{\mathbf{p}} = [\tilde{p}_1, ..., \tilde{p}_l]$ over *lost*-language characters as $\tilde{p}_j = \sum_{i=1}^{k} p_i p_{j|i}$. With this transformation in hand, we have adapted the original known-language model to both accept lost-language inputs *and* predict lost-language outputs using only the mapping α.

We now propose to fine-tune the adapted model on the *lost*-language word list, following the same procedure used to train the underlying known-language model. However, we make α the only tunable parameter, so that the only way for the model to improve the language modeling loss at this stage will be to learn the correct mappings between characters in the two scripts. This procedure makes α *trainable*, whereas the mapping in Tran 2020 was static and estimated offline.

Permutation Loss We hypothesize that α may be more easily learned if it is constrained to be approximately one-to-one, as mappings between scripts will generally be sparse. Thus we generalize the matrix penalty function from Lyu et al. (2020) to the case of non-square $k \times l$ matrices:

$$\mathcal{L}_{sparse} = \sum_{i=1}^{k} \left[\sum_{j=1}^{l} |\alpha_{ij}| - \left(\sum_{j=1}^{l} \alpha_{ij}^2 \right)^{1/2} \right] \quad (3)$$

$$+ \sum_{j=1}^{l} \left[\sum_{i=1}^{k} |\alpha_{ij}| - \left(\sum_{i=1}^{k} \alpha_{ij}^2 \right)^{1/2} \right]$$

When applied to a square matrix α, this quantity approaches zero as the matrix approaches a permutation; in the non-square setting it approaches zero as α approaches some non-square projection of a permutation. We hypothesize that tuning α to jointly minimize the sum of the cross-entropy language modeling loss with this sparsity loss will yield more accurate character-level alignments than fine-tuning on the language modeling loss by itself.

Note that, during our fine-tuning step, there is no way for a character in one script to be decoded to multiple characters in the other script; we use a vanilla Transformer architecture with no mechanism for explicit insertion or deletion operations. This creates an inductive bias towards one-to-one mappings, which is reinforced by this sparsity loss term. Despite this, we will show in Section 5.2 that our model is nonetheless capable of learning more complex, many-to-one mappings when there is evidence for such in the input data.

4. Towards Word-Level Alignment

After fine-tuning, α yields a representation for each lost-language character as a linear combination of known-language characters. We next wish to use these combinations to infer likely pairings between cognates at the word level.

4.1. Edit Distances

We first compute the Levenshtein distance (Levenshtein, 1966) from every lost word to every known word, where the cost of substituting lost character j with known character i is:

$$C(j,i) = 1 - \frac{p_{i|j}}{\max_i p_{i|j}} \quad (4)$$

where $p_{i|j}$ is the conditional probability that known character i would be sampled by lost character j paralleling Eq. (2). Note that $C(j,i) = 0$ whenever i is the nearest known-language neighbor to j in the embedding space, while for all other pairings the cost rises proportionally to the distance between i and j. Thus there is no cost to replace a lost-language character with its most likely known-language correspondent, and increasing cost for less likely substitutions.

4.2. Alignment Likelihoods

For lost- and known-language vocabularies V_l and V_k, let $\Delta \in \mathbb{R}^{|V_l| \times |V_k|}$ be a matrix where Δ_{mn} is the weighted edit distance between lost word m and known word n as described above. Let $D_m = \mathrm{softmax}(-\Delta_m)$ be a probability distribution where D_{mn} is the probability that lost word m aligns to known word n, and note that the resulting probabilities are inversely proportional to the original edit distances (Eq. (4)).

To obtain a more sophisticated model for word-level alignments, we can incorporate a prior estimate for the likelihood that known word n is cognate to some lost word, as opposed to being a distractor that has no mapping into the other language. Existing approaches to cognate detection excel in clean settings where there are few or no such distractors, so the ability to identify and prune these words would be a useful result in itself.

To this end, we propose to learn a language model on the *lost* word list, then fine-tune on the known word list following the same procedure outlined in Section 3. In the resulting model, the average perplexity when producing known word n should be low if n is cognate to some lost word, as in this case the underlying lost-language model will have seen that cognate during pretraining and the corresponding known word will look "in-domain". By contrast, if known word n is not cognate to any lost words, it should be "out-of-domain" and therefore incur a higher average perplexity. We construct a vector $\Psi \in \mathbb{R}^k$ where Ψ_n is the average perplexity when producing the characters in known word n. We convert this to a probability distribution $P = \mathrm{softmax}(-\Psi)$, and estimate the probability that lost word m aligns to known word n as $P_n^r D_{mn}$ where r is a smoothing term.

5. Experimental Results

5.1. Character Alignment

We train the proposed model on the noisy Ugaritic-Old Hebrew data from Luo et al. 2019. We consider two directions: pre-training on the known language and fine-tuning α on the lost language, and *vice versa*. In each direction, we compare models trained (i) without the permutation loss $\mathcal{L}_{sparse}$, (ii) with the permutation loss for just the first 50 iterations as a kind of warm-up, and (iii) with the permutation loss for the entire duration of training.

Pretraining Language	Permutation Loss: None top 1	None top 5	Warm-Up top 1	Warm-Up top 5	Always top 1	Always top 5
Old Hebrew (Known)	26%	57%	**83%**	**100%**	13%	30%
Ugaritic (Lost)	48%	74%	48%	70%	0%	17%

Table 1: Top-1 and top-5 precision of character-level mappings derived from α.

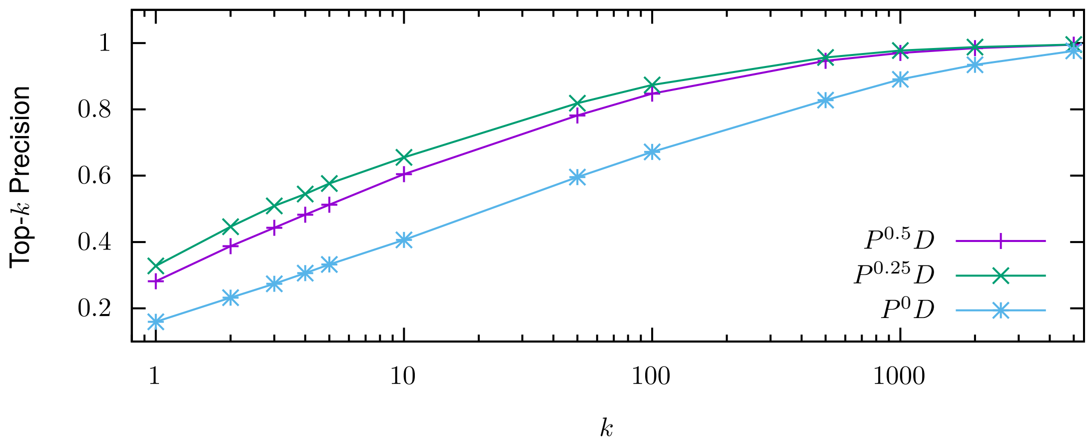

Figure 2: Top-k precision on Ugaritic-Old Hebrew cognate detection for various thresholds k and for values of the smoothing parameter $r \in [0, 0.25, 0.5]$.

α captures a nuanced mapping between characters in the two scripts in the form of a weighted sum. We attempt to concretize this into a one-to-one mapping for evaluation purposes, but note that this will necessarily lose some of the information inherent in the full set of weights. For example, Table 1 reports top-1 and top-5 precision for mappings derived by aligning each character in the known script to the character(s) with the maximmum likelihood $p_{j|i}$ according to Eq. (2).

From these results it is clear that the proposed technique can accurately learn cross-script character equivalencies, doing so most effectively when pretrained on the *known* language and fine-tuned on the lost. In this setting, α appears to perfectly capture the mapping between the two scripts, achieving 100% top-5 precision in the best case. In the opposite direction, the best result is just 74% top-5 precision. We speculate that this asymmetry derives from the fact that there is much more known- than lost-language data, meaning that the initial model will learn higher-quality character representations when pre-trained on the known language.

Our proposed permutation loss also appears to improve the character-level alignment under certain training regimes. Specifically, when pre-training on the known language, applying the permutation loss during the earliest iterations of the alignment step yields significant improvements in the quality of the learned alignment. In other settings, this term has no effect or is actively detrimental to the quality of the learned mapping. This suggests that it can be useful to "prime" the model to expect a roughly one-to-one mapping at the start of training, but that this requirement must eventually be relaxed. This makes sense given that the true mapping between these scripts includes some many-to-one or one-to-many relationships (as between Ugaritic *a, u, i* and Old Hebrew *a* in most contexts).

5.2. Word Alignment

The data contains 38898 total Hebrew words; thus *a priori* each lost (Ugaritic) word could be aligned to any of 38898 possible targets. We wish to narrow this to a small pool of candidate cognates for each word: ideally, we want just the most likely cognate in each case, but even tens of candidates per word is manageable for reranking or human evaluation.

To this end, for each lost word, we select the k known words with the highest probability according to the distribution P^rD described above. We call these *candidate cognate pairs*, and consider the alignment successful if the true cognate is among the k chosen terms. Figure 2 plots the top-k precision for different values of k and the smoothing parameter r.

Our best results arrive at $r = 0.25$, where nearly half (46%) of true cognates are either the most likely or the second-most likely candidate according to $P^r D$, and 66% fall within the top 10 most likely candidates. 66% is the same accuracy reported by Luo et al. (2019) for their noisy evaluation. These results suggest an avenue for future work whereby the top-k candidates are reranked to promote the *true* cognates to the top of the ranking. Under this approach, the correct cognate would only need to be selected from a few tens of candidates, rather than the full set of many thousands.

For additional context on these results, note that half of the word pairs in the input data have identical lengths in both languages, while the other half are longer in one language than the other. Thus any score above 50% must include some cases where the model correctly infers a complex, non-one-to-one mapping between some characters, such as the deletion of `y` from Heb `labyk` to Uga `labk` or of `awy` from Heb `wlawyb` to Uga `wlib`.

We emphasize that top-k precision is highest when we exploit P as a prior estimate for whether a word has a cognate or not. Thus P provides a usable signal to help assess whether or not a given word is a distractor. This is encouraging, as it is otherwise difficult to tell which words are "out-of-domain" when dealing with undeciphered data.

The process outlined in this work is also extremely fast, averaging just 0.015s per word, versus 0.464s per word for Luo et al. 2019 on the same data and longer still for Tamburini 2023. We are therefore optimistic that this work can serve as the basis for faster, more efficient cognate alignment models. One straightforward application of these results would be as a smart initialization for a model such as Luo et al. 2019, to bias the initial character mappings and limit the set of word-level pairings that are explored. This would increase the efficiency of such a model by constraining its search space, which we predict would also help to limit the impact of distractor vocabulary items.

6. Related Work

Aside from the works noted in Section 2, cognate prediction (Fourrier and Sagot, 2022; Fourrier et al., 2021; Hämäläinen and Rueter, 2019; Wu and Yarowsky, 2018; Dekker, 2018) is a closely-related task found in the field of historical linguistics, whereby the form of a word's cognate in some target language is predicted given that word's phonetic representation in some known language. Cognate prediction models are typically generative, producing a sequence of output phonemes given a sequence of input phonemes. Thus these models have an open output vocabulary (the set of all phoneme sequences), and their inputs and outputs come from the same script (IPA or another phonetic representation). By contrast, our model selects candidate cognates from a closed list (similarly to Beinborn et al. 2013), and assumes that inputs and outputs are written using distinct scripts. This enables the application of our model to undeciphered data, where the underlying phonetic representations are unknown, but also limits it to finding only those word pairs which are attested in the input word lists. Moreover, cognate prediction requires parallel training data, whereas our decipherment-focused approach learns from non-aligned word lists. Notably, if we omit the t-SNE inspired transformation which was used to convert model outputs from one script to the other, and omit the following edit-distance computations, our model would function as a transducer which rewrites inputs in one script into another script, in a way that would more closely resemble prior cognate prediction models. It may therefore be worth exploring applications of our model to this task in future work.

7. Conclusion

In conclusion, our work highlights the need for cognate alignment models to be both efficient and robust against noise if they are to be applied to realistic decipherment tasks. We argue that these qualities are lacking in existing approaches. As a first step towards overcoming these challenges, we have introduced a novel technique that leverages monolingual language models to swiftly and accurately learn cross-script character equivalencies. By incorporating a permutation loss term, we further improve the precision of the learned equivalencies by guiding the model towards nearly one-to-one mappings. Finally, we propose a method of combining weighted edit distances with perplexity signals which for the first time enables effective filtering of words without cognates in the paired language. These advancements lay the groundwork for the development of more efficient and reliable cognate alignment models in future work.

8. Bibliographical References

Lisa Beinborn, Torsten Zesch, and Iryna Gurevych. 2013. Cognate production using character-based machine translation. In *Proceedings of the Sixth International Joint Conference on Natural Language Processing*, pages 883–891, Nagoya, Japan. Asian Federation of Natural Language Processing.

Alexis Conneau, Guillaume Lample, Marc'Aurelio Ranzato, Ludovic Denoyer, and Hervé Jégou.

2017. Word translation without parallel data. *arXiv preprint arXiv:1710.04087*.

Peter Dekker. 2018. Msc thesis: Reconstructing language ancestry by performing word prediction with neural networks.

Chris Dyer, Victor Chahuneau, and Noah A. Smith. 2013. A simple, fast, and effective reparameterization of IBM model 2. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 644–648, Atlanta, Georgia. Association for Computational Linguistics.

Clémentine Fourrier, Rachel Bawden, and Benoît Sagot. 2021. Can cognate prediction be modelled as a low-resource machine translation task? In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 847–861, Online. Association for Computational Linguistics.

Clémentine Fourrier and Benoît Sagot. 2022. Probing multilingual cognate prediction models. In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 3786–3801, Dublin, Ireland. Association for Computational Linguistics.

Mika Hämäläinen and Jack Rueter. 2019. Finding Sami cognates with a character-based NMT approach. In *Proceedings of the 3rd Workshop on the Use of Computational Methods in the Study of Endangered Languages Volume 1 (Papers)*, pages 39–45, Honolulu. Association for Computational Linguistics.

Geoffrey E Hinton and Sam Roweis. 2002. Stochastic neighbor embedding. In *Advances in Neural Information Processing Systems*, volume 15. MIT Press.

V. I. Levenshtein. 1966. Binary Codes Capable of Correcting Deletions, Insertions and Reversals. *Soviet Physics Doklady*, 10:707.

Jiaming Luo, Yuan Cao, and Regina Barzilay. 2019. Neural decipherment via minimum-cost flow: From Ugaritic to Linear B. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3146–3155, Florence, Italy. Association for Computational Linguistics.

Jiancheng Lyu, Shuai Zhang, Yingyong Qi, and Jack Xin. 2020. Autoshufflenet: Learning permutation matrices via an exact lipschitz continuous penalty in deep convolutional neural networks. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, KDD '20, page 608–616, New York, NY, USA. Association for Computing Machinery.

Fabio Tamburini. 2023. Decipherment of lost ancient scripts as combinatorial optimisation using coupled simulated annealing. In *Proceedings of the Workshop on Computation and Written Language (CAWL 2023)*, pages 82–91, Toronto, Canada. Association for Computational Linguistics.

Ke Tran. 2020. From english to foreign languages: Transferring pre-trained language models.

Laurens van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9(86):2579–2605.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 5998–6008.

Winston Wu and David Yarowsky. 2018. Creating large-scale multilingual cognate tables. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, Miyazaki, Japan. European Language Resources Association (ELRA).

Samuel Xavier-de-Souza, Johan A. Suykens, Joos Vandewalle, and Désiré Bolle. 2010. Coupled simulated annealing. *IEEE Trans Syst Man Cybern B Cybern*, 40(2):320–335.