Estimation of Distribution Algorithms based on n-gram Statistics for Sequencing and Optimization

Chung-Yao Chuang
The Robotics Institute
Carnegie Mellon University
cychuang@cmu.edu

Stephen F. Smith
The Robotics Institute
Carnegie Mellon University
sfs@cs.cmu.edu

ABSTRACT

This paper presents our work on Estimation of Distribution Algorithms (EDAs) that address *sequencing problems*, i.e., the task of finding the best ordering of a set of items or an optimal schedule to perform a given set of operations. Specifically, we focus on using probabilistic models based on n-gram statistics. These models have been used extensively in modeling the statistical properties of sequences. We start with an EDA that uses a bigram model, then extend this scheme to higher-order models. However, directly replacing the bigram model with a higher-order model results in premature convergence. We give an explanation on this situation, along with some empirical support. We then introduce a technique for combining multiple models of different orders, which allows for smooth transition from lower-order models to higher-order ones. Furthermore, this technique can also be used to incorporate other heuristics as well as prior knowledge about the problem into the search process. Promising preliminary results on solving Traveling Salesman Problems (TSPs) are presented.

Categories and Subject Descriptors

I.2.8 [**Artificial Intelligence**]: Problem Solving, Control Methods, and Search—*Heuristic methods*

1. MODELING WITH N-GRAM STATISTICS

An n-gram is a pattern of n consecutive items, which is usually a segment from a longer sequence. Such a construct is often used in the field of natural language processing (NLP). For example, a classic task in NLP is to estimate the conditional probability of observing some item (word) w_i as the next item, given the history of items (words) seen so far. Specifically, we are interested in estimating

$$P(W_i = w_i | W_{i-n+1} = w_{i-n+1}, \ldots, W_{i-1} = w_{i-1})$$

where the sequence $w_1, w_2, \cdots$ is some instantiation of a sequence of random variables $W_1, W_2 \cdots$. In the following, we will use $P(w_i|w_{i-n+1} \cdots w_{i-1})$ as a shorthand.

GECCO'14, July 12–16, 2014, Vancouver, BC, Canada.
ACM 978-1-4503-1964-5/14/07.
http://dx.doi.org/10.1145/2598394.2598399 .

The obvious first answer to the above formulation is to suggest using a *maximum likelihood estimate* (MLE):

$$P_{\mathrm{MLE}}(w_i|w_{i-n+1} \cdots w_{i-1}) = \frac{C(w_{i-n+1} \cdots w_{i-1} w_i)}{\sum_{v \in \mathbf{V}} C(w_{i-n+1} \cdots w_{i-1} v)}$$

where $C(w_{i-n+1} \cdots w_i)$ is the frequency of a certain n-gram in training samples, and $\mathbf{V}$ is the set of possible items. However, a drawback is that MLE assigns a zero probability to unseen events, which effectively zeros out the probability of sequences with component n-grams that just happened not appearing in the training samples. For our scenario, this creates a risk of arbitrarily discarding some portion of the unexplored search space. A simple solution used in this work is to smooth the distribution with some *pseudocount* κ:

$$P_\kappa(w_i|w_{i-n+1} \cdots w_{i-1}) = \frac{C(w_{i-n+1} \cdots w_{i-1} w_i) + \kappa}{\sum_{v \in \mathbf{V}} (C(w_{i-n+1} \cdots w_{i-1} v) + \kappa)}$$

where κ is usually set to a value smaller than 1.

2. USING N-GRAM MODELS IN EDAS

In this work, instead of generating an entire solution anew, we first take an existing solution from current population and randomly extract a subsequence from it. This segment will then be taken as the starting point of new solution and serve as the "history" on which further sampling is based. This kind of *partial sampling* technique has been used previously [2, 4], achieving better usage of diversity. For our purpose, this has an additional benefit of providing a convenient basis to initialize the sampling from n-gram models.

To summarize the overall flow of the algorithm: At each iteration, we first estimate an n-gram model from the current population. To generate a new solution, we use partial sampling on an existing solution in the current population. Each solution in the current population is visited once for such sampling. Following each partial sampling, a replacement competition is hold between the new solution and the one from which the starting segment was extracted. Note that we use replacement as the sole means for selecting promising solutions, i.e. better solutions are preserved under the replacement process. This is similar to the evolution strategy, in which every solution in the current population is seen as potentially good solution because they have survived previous replacement competitions.

As a first step, we experimented with a bigram model,

$$P_{2\mathrm{G}}(w_i|w_{i-1}) = \frac{C(w_{i-1} w_i) + \kappa}{\sum_{v \in \mathbf{V}} (C(w_{i-1} v) + \kappa)}$$

for solving a 48-city TSP, `gr48`*. Let ℓ denote the problem size. The population size N is set to 5ℓ, the pseudocount

*`http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/`

Table 1: Preliminary Experiments on gr48

Method	Success Rate	# of Evaluations mean	# of Evaluations std
2G	30/30	277024	34535.4
3G	9/30	224640	118490.2
2G $\xrightarrow{800\ \text{iter.}}$ 3G	30/30	240032	20753.3
2G+3G	30/30	230048	23985.8
2G+3G+DH	30/30	154984	20243.5

$\kappa = 0.01$, and termination criterion is when either the optimal tour is found or when the algorithm reaches 50ℓ iterations. The result of the experiment is presented in Table 1. It shows the success rate in finding the optimal tour and the average number of function evaluations used among the successful runs. It can be seen that the bigram approach gives a pretty decent performance. A tempting thought to proceed is to directly replace the bigram with a trigram

$$P_{3G}(w_i|w_{i-2}w_{i-1}) = \frac{C(w_{i-2}w_{i-1}w_i) + \kappa}{\sum_{v\in\mathbf{V}}(C(w_{i-2}w_{i-1}v) + \kappa)}$$

for learning the patterns, but this results in a significant drop in success rate. Our explanation is that at early stage of a run, there are not so many long patterns that are of good quality. If we attempt to use a higher-order model to learn longer patterns when there are none, we will end up encoding mediocre patterns into the model. To provide some empirical support, we modified the process to begin with a bigram and switch to trigram after 800 iterations. As shown in Table 1, this remedies the problem. However, choosing an adequate switching point is nontrivial task. To address the issue, we propose combining multiple models of different orders and use a set of weights to control the emphasis.

3. COMBINING MULTIPLE MODELS

Specifically, we formulate the synthesis of K models as

$$P(w_i|\mathbf{h}_i) = \sum_{j=1}^{K} \lambda_j P_j(w_i|\mathbf{h}_i) \quad (1)$$

where $\mathbf{h}_i$ represents the history of items seen so far and λ_j is the weight associated with the jth model s.t. $\lambda_j > 0$ and $\sum_j \lambda_j = 1$. A combination of bigram and trigram will be

$$P_{2G+3G}(w_i|\mathbf{h}_i) = \lambda_{2G}P_{2G}(w_i|w_{i-1}) + \lambda_{3G}P_{3G}(w_i|w_{i-2}w_{i-1})$$

Assuming that we want to combine K models, then we have to determine K weights, $\lambda_1, \lambda_2, \ldots, \lambda_K$, associated with those models. To do this, we reserve a portion of the population for the task of estimating those λ_j's. Suppose that there are M items in such a holdout set for which we can give conditional probabilities. For each model P_j, we create a stream $\mathbf{p}_j = (p_{j1}, p_{j2}, \ldots, p_{jM})$ where p_{ji} is the probability of item w_i predicted by the model P_j, i.e., $p_{ji} = P_j(w_i|\mathbf{h}_i)$. These K probability streams (each of length M) are then

Algorithm 1 Estimating the Weights λ_j's

Input: a set of probability streams $\{\mathbf{p}_j\}$ where $j = 1$ to K. Each $\mathbf{p}_j = (p_{j1}, p_{j2}, \ldots, p_{jM})$ is of length M.

Initialize $\mathbf{\Lambda}^{(0)} = \{\lambda_j^{(0)}\}$ s.t. $\lambda_j^{(0)} > 0$ and $\sum_{j=1}^{K} \lambda_j^{(0)} = 1$

repeat

$j = 1 \ldots K$, update: $\lambda_j^{(t+1)} = \frac{1}{M}\sum_{i=1}^{M} \frac{\lambda_j^{(t)} p_{ji}}{\sum_{k=1}^{K} \lambda_k^{(t)} p_{ki}}$

$t = t + 1$

until the difference between $\mathbf{\Lambda}^{(t)}$ and $\mathbf{\Lambda}^{(t-1)}$ is small

return $\mathbf{\Lambda}^{(t)}$

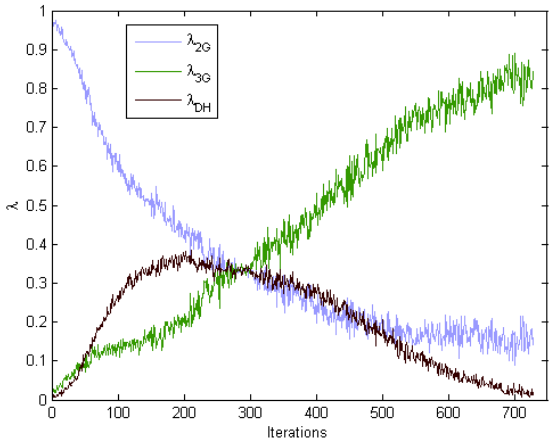

Figure 1: Weight variation: combining of bigram, trigram and distance heuristics for solving gr48.

Table 2: Performance Comparison on pr76

Method	N	Success Rate	# of Evaluations mean	# of Evaluations std
OX	960	0/10	N/A	N/A
eER	960	3/10	394887	22321
PMX	960	0/10	N/A	N/A
EHBSA-WT	120	9/10	457147	65821
2G+3G	120	10/10	405660	54893
2G+3G+DH	120	10/10	195960	28123

used as the input to Algorithm 1 [3]. The resulting λ_j's optimize the average likelihood w.r.t. this holdout set.

Note that this technique can also be used to incorporate other heuristics into the search. For example, if we want to add a distance-based heuristic for TSP into the search mechanism, we could do so by crafting an "artificial distribution"

$$P_{DH}(w_i|w_{i-1}) = \frac{(d(w_{i-1}, w_i))^{-10}}{\sum_{v\in\mathbf{V}}(d(w_{i-1}, v))^{-10}}$$

where $d(u, v)$ is the distance between city u and city v. This assigns a larger probability mass to a city that has shorter link to the last city in the partial tour constructed so far.

To illustrate the search behavior, Figure 1 shows the variation of weights in a typical run that uses a combination of P_{2G}, P_{3G} and P_{DH}. The performance of these methods on gr48 are also shown in Table 1 (with 10% of the population as the holdout set.) To provide some comparison to other approaches, we adopt the results of [4] on pr76, a 76-city TSP. Table 2 lists their method, EHBSA-WT, along with 3 classical GA approaches, OX, eER, and PMX. Note that EHBSA-WT gave the best empirical performance in [1].

4. REFERENCES

[1] J. Ceberio et al. A review on estimation of distribution algorithms in permutation-based combinatorial optimization problems. *Progress in Artificial Intelligence*, 1(1):103–117, 2012.

[2] C.-Y. Chuang and Y.-p. Chen. On the effectiveness of distributions estimated by probabilistic model building. In *Proceedings of GECCO-2008*, pages 391–398.

[3] F. Jelinek and R. L. Mercer. Interpolated estimation of markov source parameters from sparse data. In *Pattern Recognition in Practice*, 1980.

[4] S. Tsutsui, M. Pelikan, and D. E. Goldberg. Using edge histogram models to solve permutation problems with probabilistic model-building genetic algorithms. Technical Report 2003022, 2003.