Combinatorial Optimization EDA using Hidden Markov Models

Marc-André Gardner, Christian Gagné, and Marc Parizeau
Laboratoire de vision et systèmes numériques
Département de génie électrique et de génie informatique
Université Laval, Québec (Québec), Canada G1V 0A6
marc-andre.gardner.1@ulaval.ca, {christian.gagne, marc.parizeau}@gel.ulaval.ca

ABSTRACT

Estimation of Distribution Algorithms (EDAs) have been successfully applied to a wide variety of problems. The algorithmic model of EDA is generic and can virtually be used with any distribution model, ranging from the mere Bernoulli distribution to the sophisticated Bayesian network. The Hidden Markov Model (HMM) is a well-known graphical model useful for modelling populations of variable-length sequences of discrete values. Surprisingly, HMMs have not yet been used as distribution estimators for an EDA, although they are a very powerful tool for estimating sequential samples. This paper thus proposes a new method, called HMM-EDA, implementing this idea, along with some preliminary experimental results.

Categories and Subject Descriptors

I.2.8 [**Artificial Intelligence**]: Problem Solving, Control Methods, and Search—*Heuristic methods*; G.3 [**Probability and Statistics**]: *Markov processes*

Keywords

Estimation of distribution algorithms; Hidden Markov models; Combinatorial optimization

1. INTRODUCTION

Combinatorial optimization problems are of high interest in the evolutionary computing community. Besides their various real-world applications, they are often very hard to solve in an efficient manner: most of these problems are NP-complete, and the interactions between their variables make their modelling difficult.

An Estimation of Distribution Algorithm (EDA) is an optimization method where the search of an optimum is led by an explicit probabilistic model [5]. This probabilistic model may be of various kinds, depending on the targeted problem. In the specific case of combinatorial optimization, and especially permutation problems, some EDAs have already been proposed. Among them, we may mention the ICE [3] method, which makes use of a standard GA where the crossover points are selected through the underlying EDA, EHBSA (Edge Histogram Based Sampling Algorithm) [8], which stores information on the permutations in an Edge Histogram Matrix, and the *dependency-tree EDA* (dtEDA) [7]. A dependency tree is similar in some ways to a Bayesian network as it is able to find and learn the dependencies between the vector components. However, Hidden Markov Models (HMMs) have never been applied in an EDA perspective, even if their underlying sequence nature makes them very suitable for the modelling of combinatorial optimization problems. In this paper, we are proposing a method called HMM-EDA, which is a proof-of-concept of the applicability of HMMs for modelling EDAs.

Copyright is held by the author/owner(s).
GECCO'13 Companion, July 6–10, 2013, Amsterdam, The Netherlands.
ACM 978-1-4503-1964-5/13/07.

2. HIDDEN MARKOV MODELS

Before going into design and implementation details of HMM-EDA, a brief review of some important notions on Markov models is presented below.

2.1 Markov Process

A Markov process is a stochastic process with a conditional probability distribution that satisfies the *Markov property*. This property specifies that all the information needed to obtain the probability distribution of the next state can be retrieved in the current state, that is to say that the other past states have no influence at all on the distribution. In a discrete distribution context, this property is often formalized as:

$$\begin{aligned} P(X_t = x_t | X_{t-1} = x_{t-1}, X_{t-2} = x_{t-2}, ...) \\ = P(X_t = x_t | X_{t-1} = x_{t-1}) \end{aligned} \quad (1)$$

A Markov process is thus frequently said to have no *memory*, since transitions between its states depends only on the current state of the process. This property drastically reduces the model complexity, and thus the requirements in terms of computational effort and memory used.

2.2 Markov Chain

A Markov chain is a simple yet interesting example of a Markov process. It is characterized by a set of n discrete states, and by a $n \times n$ *transition matrix*. This matrix specifies the transition probabilities from one state to another: the value at position (i, j) in the matrix is the probability to transition to state j when the current state is i. By definition, lines of the transition matrix sum to 1. The diagonal values of the matrix provide the self-transition probabilities, that is the probability that the process stays in the same state at $t + 1$.

While very interesting in various ways, the Markov chains are restricted to the modelling of phenomena which also respect the Markov property. This assumption is rarely met

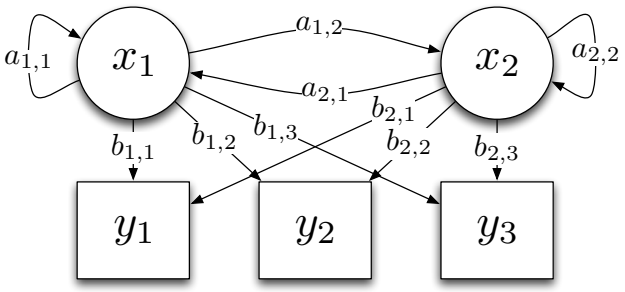

Figure 1: Depiction of a HMM with two hidden states (x_i), the associated transition probabilities ($a_{i,j}$), three possible observations (y_i), and their corresponding emission probabilities ($b_{i,j}$).

in real-world problems, or even in some classical learning problems. For instance, in the Travelling Salesman Problem (TSP), the choice of the next city to visit is not dependent only on the previously visited one, but on all the cities visited so far. This limitation makes the Markov chains unsuitable for many applications.

2.3 Hidden Markov Models

A hidden Markov model (HMM) can be seen as a Markov chain for which the current state cannot be directly observed (e.g. it is *hidden*). Instead, each of the states of the Markov chain *emits* an observation from a previously defined set. This emission process is also stochastic, that is a given state only specifies a probability distribution over the observations set. Accordingly, the same observation may be generated by more than one state.

These hidden states still rely on the Markov property. However, the global stochastic process is no longer limited to the modelling of sequences following the Markov property, since the hidden states provide the *memory* needed to learn more complex, arbitrary sequences.

An HMM has thus N hidden states and M observations, respectively $\{x_1, x_2, \ldots, x_N\}$ and $\{y_1, y_2, \ldots, y_M\}$. It is fully described by parameters set $\theta = \{\boldsymbol{\pi}, \boldsymbol{A}, \boldsymbol{B}\}$, where:

- $\boldsymbol{\pi}$ is **an initial probability vector** of size N, that gives the probability π_i of starting the HMM in a state x_i;
- $\boldsymbol{A}$ is **a transition probability matrix** of size $N \times N$, that gives the probability $a_{i,j}$ of moving into state x_j at the next time step when the current state is x_i;
- $\boldsymbol{B}$ is **an emission probability matrix** of size $N \times M$, that gives the probability $b_{i,j}$ of making the observation y_j when the state is x_i.

Fig. 1 illustrates an HMM of two states and three possible observations, along with their transition and emission probabilities. Notice how each observation (y) has no direct dependency on other ones.

A generic learning process for those parameters has been proposed through the *Baum-Welch algorithm*. This algorithm is an instance of the general Expectation Maximization procedure (EM), and takes a set of sequences of observations. It works by increasing the generation probability of these sequences through the HMM, and it has been demonstrated that the value of this generation probability will be at least equal to the value before the application of Baum-Welch [1, 6]. However, as every EM variant, it is quite sensitive to its initial parameter values.

3. ESTIMATION OF DISTRIBUTION ALGORITHMS

A general model of Estimation of Distribution Algorithms (EDAs) [5] consists in iteratively generating samples (solutions) from the current estimated distribution, evaluating these samples according to a problem-specific fitness function, and then updating the estimation of the distribution from the best samples generated. Different EDAs are mainly characterized by the type of distribution model they use, and the distribution model should itself be carefully chosen according to the type of problem that needs to be solved.

3.1 Proposed method

The proposed approach integrates an HMM in the generic EDA model, using it to directly estimate the distribution of the samples making up the population. Using an HMM for that purpose is quite versatile, as it only assumes that the samples are sequences of discrete values. Such a model should be useful in a wide variety of optimization problems, including problems where solutions can be modelled as bit strings. HMM should be able to capture complex interactions between the elements of sequences, not only the first order relations captured in observable Markov models.

The application to the EDA process is quite straightforward. Indeed, we can directly map each possible value to an observation of the HMM. For example, in a features selection problem, we would have as many observations as the number of available features. As a result, the HMM training process can be directly fed with a given set of individuals (for instance the top 10% of the population in terms of fitness), and the individuals generation process boils down to the sampling of the trained HMM.

The production of individuals of variable length is not a problem for the HMM, since we can sample the model as much as we want. In the same way, the generation of a fixed length sequence is quite easy – we simply have to place a hard length limit over the sampling process. However, in the case of permutation problems, an issue arises, as each value must be generated only once in an individual. For instance, in the TSP, each town must be visited once and only once. To avoid the generation of the same value twice in an individual, we have to place some additional restrictions over the HMM used. Those restrictions are implemented by adding a binary mask over the *emission probability matrix* $\boldsymbol{B}$. Initially, all the values of this mask are set to 1, that is, every emission may be selected. When an observation is made, its corresponding probability is set to 0 *for every state*, and the remaining observation probabilities are normalized.

The method is based on a $(\mu + \lambda)$ selection loop, with the addition of the model training and individual sampling processes. The population is first initialized as in a GA, and the fitness of each individual is computed. Then, while a stopping criteria is not met (in this case, a maximum number of iterations), the optimization proceeds with:

1. Tournament selection to select the most valuable individuals of the population. This selection operator

implies that a given individual might be selected several times. This is intended, as those multiple copies will bias the HMM toward learning more accurately the best individuals of the pool.

2. HMM parameters ($\theta = \{\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{\pi}\}$) initialization using normalized uniformly distributed values, to ensure that related probabilities sum up to 1. Initial randomness is important to facilitate the convergence of the Baum-Welch algorithm.

3. HMM training with the individuals selected at step 1, using the Baum-Welch algorithm.

4. Offspring population generation, directly using the HMM output as new samples. In the generation process, some emissions can be randomized, as explained below. Between each generation of a sample, the HMM is reset, i.e. the sequence corresponding to a generated individual does not depend on the previous samples.

5. $(\mu + \lambda)$ selection, checking to select at most one copy of duplicated individuals in the new population.

The proposed method makes use of some mutation when generating individuals with the HMM. It consists in modifying the usual mechanism of observation emission, such that, according to a given mutation probability, an observation is selected randomly (uniformly) instead of using the HMM emission probabilities of the current state. This operation should allow exploration and diversity to be increased, by producing some random elements in some of the sequences. Preliminary experiments suggested that a mutation probability around 2% for each emission is in general a good parametrization.

4. EXPERIMENTS

As preliminary experiments, the proposed approach has been tested on two classical problems, that is the *Travelling Salesman Problem* and a *Features Selection problem*, where the algorithm has to choose a given number of features of a classifier in order to maximize the classification rate.

For both problems, we compare the results of HMM-EDA to a classical GP approach. In the case of the features selection problem, we also added a simple EDA, PBIL [2], which works with fixed-size bit string samples and assumes a multivariate Bernoulli distribution of these samples. While very simple, this EDA will characterize the level of performance which could be achieved by an algorithm which does not take interactions between variables into account.

All experiments were conducted with the DEAP (Distributed Evolutionary Algorithms in Python) framework[1]. The implementation of the HMM for the HMM-EDA relies on the GHMM library[2].

4.1 Features Selection for Classification

For the experiment, we used the *spambase* dataset, available on the UCI machine learning repository[3]. It is a dataset made of 4601 different instances, each of these being an email categorized into two classes, spam or non-spam, with 57 available features. The features are mostly the frequency of specific words in the email. Given that not all words are useful to categorize an email as spam, it makes sense to apply a features selection methods to this dataset. For each run, the set has been randomly partitioned in a training (33% of the instances) subset and a testing (67%) subset. For all 100 runs made with HMM-EDA, GA, and PBIL, the corresponding run of each method was conducted using the same train/test partition. Features selection is conducted following a wrapper model [4], using a nearest neighbour as classifier, as it requires no extensive training. The feature selection problem is to select the 8 best features among the 57 available.

To produce comparable results, we used a different representation for each technique: generating a sequence of 8 features for the HMM-EDA approach, using the 8 first elements of a permutation vector for the GA, and making use of a bit string of length 57 with PBIL, using the 8 first features for which the bit is set to 1 in the string.

As for the parametrization, all methods were granted with a total of 20 000 evaluations and used a tournament selection of size $n = 5$. GA and PBIL used a population size of 500. Also, GA used a crossover/mutation rate of 0.3 and 0.2, while HMM-EDA used a $(\mu + \lambda)$ algorithm with $\mu = 250$ and $\lambda = 750$, and a mutation rate of 0.02. The number of hidden states was set to 10, since it was empirically found to allow a sufficient model complexity without compromising its generalization ability. The learning rate and mutation probability of PBIL were set to 0.25 and 0.1 respectively.

Results obtained (Fig. 2 and 4) show that the final fitness achieved by both HMM-EDA and GA is the same, but HMM-EDA converge slightly faster to the optimal solution. The classification rate reached (about 90%) is good and consistent with the current best classification rates for this dataset, which is about 93%, but with the use of all features. PBIL results are quite interesting, because they clearly show that the combinations of features can have a considerable impact on the classifier performance. For example, two variables taken individually may have little impact when selected as a feature on the performance of the classifier, but when selected together, the performance of the classifier might be improved. This hypothesis is verified by the poor performance of PBIL on this problem. Besides, HMM-EDA performance on this problem indicates a good versatility and an implicit adaptability. In other words, relations between variables can be learnt by HMM-EDA, but the model will not try to learn relations when there are none, which are both important characteristics for a successful adaptive optimization method.

4.2 Travelling Salesman Problem

The Travelling Salesman Problem (TSP) is another classical optimization problem. The general problem is defined as the search of the shortest Hamiltonian circuit in a fully connected graph. For GA and HMM-EDA, a potential solution is represented as a vector of indices, each index designating a city to visit. The order in which the indices are organized establishes the order of traversal. For HMM-EDA, observations corresponding to cities that have already been visited are masked, as explained in the third paragraph of Sec. 3.1. The same starting point was used for every evaluated individual. Both GA and HMM-EDA were granted with 60 000 evaluations and used a tournament selection of size $n = 4$. GA used a population size of 500 and a crossover/mutation

[1] `http://deap.gel.ulaval.ca`
[2] `http://ghmm.org`
[3] `https://archive.ics.uci.edu/ml/datasets/Spambase`

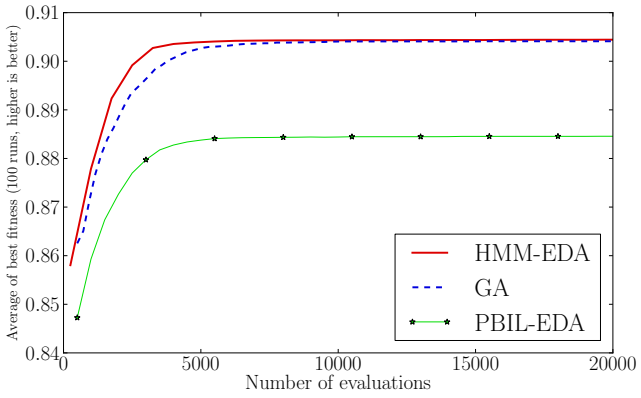

Figure 2: Evolution of the fitness against the number of evaluations performed for Features Selection.

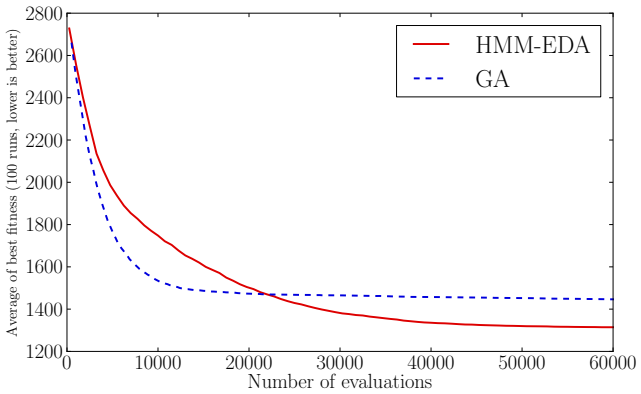

Figure 3: Evolution of the fitness against the number of evaluations performed for TSP.

rate of 0.3 and 0.2, while HMM-EDA used a $(\mu + \lambda)$ algorithm with $\mu = 250$ and $\lambda = 750$, and a mutation rate of 0.02. The number of hidden states was set to 25, as the highly interdependent values require a much more complex model. PBIL has not been applied to this problem as the bit string representation is not suitable for that purpose.

In our experiments, we used a pre-computed graph dataset containing 24 cities. The optimal solution is known to be of length 1272. The same dataset was used over the 100 runs.

In this problem (see Fig. 3), GA converges faster than HMM-EDA, but its final fitness value is significantly worse than the one obtained by HMM-EDA. As the box-plot (Fig.4) clearly shows, the latter is often even able to reach the optimal path. Moreover, the variance on the GA results is almost three times that of HMM-EDA, indicating that the HMM approach is stable over its runs.

5. CONCLUSION

In this paper, we introduced HMM-EDA, a new EDA based on Hidden Markov Models. Experiments have been made over two classical problems, with results showing that the proposed approach is indeed promising: it achieves similar if not better performance than a standard permutation GA and converges more quickly to a good solution. Moreover, it can be easily adapted to virtually any problem with a

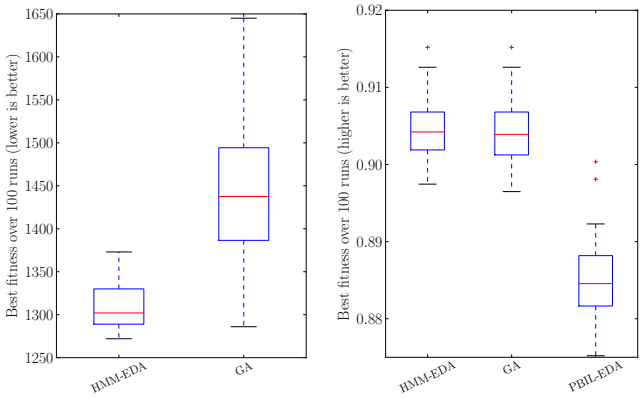

Figure 4: Box plots of the final best fitnesses (left : TSP, right : Features Selection)

representation based on a finite alphabet, and can be tuned easily by changing only one parameter (that is, the number of hidden states). Therefore, HMM-EDA might be of great interest, especially where other EDAs are likely to fail or perform poorly.

Acknowledgements

This work has been made possible through funding from NSERC (Canada) and access to computational resources of Calcul Québec / Compute Canada. We also thank Annette Schwerdtfeger for proofreading the manuscript.

6. REFERENCES

[1] E. Alpaydin. *Introduction to Machine Learning*. MIT Press, 2nd edition, 2010.

[2] S. Baluja. Population-based incremental learning: a method for integrating genetic search based function optimization and competitive learning. Technical Report CMU-CS-94-163, Carnegie Mellon University, Pittsburgh, PA, 1994.

[3] P. Bosman and D. Thierens. Crossing the road to efficient IDEAs for permutation problems. In *Proc. of the Genetic and Evolutionary Computation Conference (GECCO)*, 2001.

[4] I. Guyon and A. Elisseeff. An introduction to variable and feature selection. *Journal of Machine Learning Research*, 3:1157–1182, 2003.

[5] M. Hauschild and M. Pelikan. An Introduction and Survey of Estimation of Distribution Algorithms. *Swarm and Evolutionary Computation*, 1(3):111–128, 2011.

[6] K. P. Murphy. *Machine Learning: A Probabilistic Perspective*. MIT Press, 2012.

[7] M. Pelikan, S. Tsutsui, and R. Kalapala. Dependency trees, permutations, and quadratic assignment problem. In *Proc. of the Genetic and Evolutionary Computation Conference (GECCO)*, 2007.

[8] S. Tsutsui. Probabilistic model-building genetic algorithms in permutation representation domain using edge histogram. In *Parallel Problem Solving from Nature (PPSN)*, 2002.