Particle Swarm Optimisation of Spoken Dialogue System Strategies

Lucie Daubigney[1,3], *Matthieu Geist*[1], *Olivier Pietquin*[1,2]

[1]IMS-MaLIS – Supélec (Metz, France)
[2]UMI 2958 – GeorgiaTech/CNRS (Metz, France)
[3] Team project Maia – Loria (Nancy, France)

firstname.name@supelec.fr

Abstract

Dialogue management optimisation has been cast into a planning under uncertainty problem for long. Some methods such as Reinforcement Learning (RL) are now part of the state of the art. Whatever the solving method, strong assumptions are made about the dialogue system properties. For instance, RL assumes that the dialogue state space is Markovian. Such constraints may involve important engineering work. This paper introduces a more general approach, based on fewer modelling assumptions. A Black Box Optimisation (BBO) method and more precisely a Particle Swarm Optimisation (PSO) is used to solve the control problem. In addition, PSO allows taking advantage of the parallel aspect of the problem of optimising a system online with many users calling at the same time. Some preliminary results are presented.

Index Terms: spoken dialogue system, black-box optimisation, dialogue management

1. Introduction

Spoken Dialogue Systems (SDS) are now powerful tools to complete various tasks. Examples are booking train or flight tickets, scheduling appointments, asking for tourist information, *etc*. During the dialogue, at each turn of the system, the Dialogue Manager (DM), which is the decision maker component of the SDS, has to choose what to say next to the user, such that, at the end of the dialogue, the user's request is fulfilled. The DM has to find the sequence of dialogue acts which leads to solve efficiently the task. The dialogue management problem is thus a sequential decision problem. Since the system has to face a real user, its behaviour is expected to be as consistent as a human's behaviour would be. Consequently, the system must exhibit tailored dialogue strategies in order not to bore the user. Solving the problem gets more complicated because of the variability between the user's behaviour and the uncertainty introduced by the speech and semantic analysers, two of the most important modules of an SDS. Indeed, the modules are error-prone and some misunderstanding during the recognition of the user act may appear.

Developing such strategies is not obvious. For example, strategies have been first defined by means of hand-coded rules [1] or by means of finite-state machines [2]. The dialogue strategy is represented in a graph. Each node of the graph is a dialogue act. The transition to go from one node to another is defined by the designer of the system. The number of different situations the DM is able to face is thus limited. Because of the intractability of hand-coded strategies when the dialogue task becomes realistically complex, automatic methods coming from Artificial Intelligence and Machine Learning have been developed. First, planning algorithms [3] were proposed. Yet, planning makes a lot of assumptions such as being able to enumerate all the possible contexts or knowing transition probabilities between states given actions. Also, the objective has to be known in advance so that the optimal path in the graph can be computed. Once the plan is computed, it cannot be modified even though the interaction goes wrong.

Planning under uncertainty is thus mandatory to take the possible failures into account. For this reason, the Markov decision theory framework [4] was proposed for formulating and solving such problems [5]. The dialogue management problem has been cast into a Markov Decision Process (MDP) [6] and Reinforcement Learning (RL) can be used to find an optimal policy. Within this framework, the quality of each interaction between the user and the DM is quantified thanks to a numerical value called a *reward*. This quantity can be measured for example through the user satisfaction at the end of the dialogue, the completion of the task, the time needed to complete it, or by using a combination of several values [7]. The aim is to find the controller, which is in charge of associating to each encountered situations (dialogue contexts) a DM action, that maximises the cumulative rewards.

The Reinforcement Learning (RL) framework [8] has been proven efficient to solve MDPs when the model of transition from one state to another is unknown. This method has been applied to dialogue management in [9, 10, 11]. But, once again, this framework makes several strong assumptions. For instance, the dialogue contexts cannot be perfectly observed due to the recognition error introduced by the speech and the semantic analysers. The task is therefore non-Markov in the observation space. To meet the Markov assumption made by the MDP framework, the underlying states have to be inferred from observations using what is called a belief tracker. For example, the *Hidden Information State* [12] paradigm builds a list of the most probable current situations given the past observations, which is supposed to be a Markovian representation allowing for taking decisions in the MDP framework. .

Finally, to take into account the perceptual aliasing problem introduced by error-prone speech and language understanding modules, Partially Observable MDP (POMDP) have been proposed to model the dialogue management task [13]. Yet, solving the POMDP problem requires the transition and observation models to be known which also requires a lot of assumptions and engineering work.

In this paper, we propose to adopt a Black Box Optimisation (BBO) point of view to solve the strategy optimisation problem with fewer assumptions. This method is usually used to solve general optimisation problems. Its quality relies on the fact that, contrary to gradient methods, no strong hypotheses about the function to optimise is required (such as differentiability) and that the optimisation process does not stay stuck into local optima. The BBO finds the optimal solution by iteratively testing a set of candidate solutions in the search space. Each one of them is called a *particle*. The only information the BBO needs to perform the optimisation is an evaluation of the quality of each of the possible solutions. By means of this information, the best candidates of each turn are retained for the next one. The selection methods depend on the type of BBO algorithm used. Iterations are repeated until some global criterion is met, such as a given number of iterations is reached, or the fitness function of the best particle has reached a given value.

In the dialogue management case, each candidate strategy can be evaluated by computing a score while tested on users. The score can be the cumulative rewards for example or a subjective score provided by the user after an interaction. BBO algorithms have already been applied to solve control under uncertainty problems, such as Covariance Matrix Adaptation-Evolution Strategies (CMA-ES) [14, 15] or cross entropy methods [16, 17]. In the dialogue case, the evaluation of the fitness is the quality of a whole dialogue. It has to be noticed that this information is much less informative than the reward given at each turn in the Markovian RL framework. Here, the Particle Swarm Optimisation (PSO) method is chosen [18, 19].

At each turn of the BBO algorithm, several strategies are tested at the same time. This parallel architecture of the algorithm particularly fits for DM optmisation. Indeed, several users may call at the same time. Instead of having all of them interact with a unique strategy currently learnt (which is the case for previous solutions proposed [20, 21, 22]), several users test different candidate strategies while all the rest of users are interacting with the best one learnt so far. In consequences, the convergence rate towards the optimal solution might be increased in terms of time duration (maybe not in terms of dialogues) and fewer users might be annoyed by poor policies in the early stages of learning.

The paper is organised as follows. In Sec. 2, an overview of the PSO algorithm is presented so as its application to the DM framework. In Sec. 3 are presented the experimental settings to illustrate the method and finally, in Sec. 4 are presented some results about the test of the method on a spoken dialogue system.

2. Black Box and DM optimisations

2.1. Criterion to optimise

The general optimisation problem to be solved is to find the strategy, called a *policy*, which maximises a score related to the quality of a dialogue. A policy is a mapping from the *state* space S to the *action* space A, $: S \to A$. The state space includes all the dialogue contexts the dialogue manager is able to handle. One has to remind that some recognition error might be introduced by the speech and semantic analysers thus the real state of the dialogue is not perfectly known. It can also be a continuous space which makes the exhaustive listing of context impossible. Usually, the current state is built to be a summary of the situations and actions previously encountered. Here, it is built from the state returned by the *Hidden Information State* (HIS) [12]. The action space consists of all the actions the DM can perform, such as: "asking for information", "providing information", *etc*.

The goal of the optimisation problem is to find the strategy * which maximises some criterion J which quantifies the performance of the policy: $^* = \arg\max_{\pi:S\to A} J^\pi$: The criterion is related to the quality of a dialogue and defined here as follows:

$$J^\pi = E\left[20 \cdot \ _{\text{fulfil}} - N_{\text{turns}}\right]; \quad (1)$$

with $_{\text{fulfil}}$ equal to 1 if the task has been completed at the end of the dialogue, 0 otherwise and N_{turns} standing for the length of the dialogue.

A parametric policy is defined, $_\theta(s)$, $\in \mathbb{R}^n$ being a vector of $n \in \mathbb{N}$ parameters. The optimisation solution thus reduces to find the optimal vector * associated with the optimal strategy: $^* = \arg\max_\theta J^{\pi_\theta}$.

2.2. Particle Swarm Optimistion

PSO is a BBO algorithm inspired by methods aiming at modelling the general behaviour of a bird flock or a fish school. It is a biologically-inspired algorithm that searches for basic rules defined for each agent of the flock which can explain the emergence of a coherent group behaviour. The rules are related to the position, the velocity and the neighbourhood of each of the agents. Each move of the agent impacts on its neighbours according to the rules and the whole flock is possibly disturbed.

During the modelling, an optimisation can be performed provided that the position of each of the agent can be quantified by means of a score. If the space where the flock can possibly move is considered as the search space for the optimisation problem solutions and each agent is considered as a candidate solution, finding the solution thus reduces to find the agent which maximises the score. This is done by iteratively selecting the best agent at each time step and by moving the flock towards it and towards the best one ever encountered until some criterion is reached. This approach has been first developed by [18, 19] for solving optimisation problems. Moreover, this method has been proven efficient on solving problems considered as benchmark problems in RL in [23].

The flock of elements is called a *swarm* and each of the agents is called a *particle*. Here, a standard implementation is chosen [24]. The swarm used in this article contains N_{PART} particles with a von Neumann topology. The rules to update the velocity v^j and the position p^j of a particle j at time step i are the following:

$$\begin{aligned} v_{i+1}^j &= wv_i^j + c_1 r_1 \cdot (b^j - l^j) + c_2 r_2 \cdot (l^j p^j) \\ p_{i+1}^j &= p_i^j + v_{i+1}^j \end{aligned}$$

with some constant parameters w, $c1 = c2$, r_1 and r_2, b^j the best position ever found by the particle j and l^j the best position ever found by one particle in the neighbourhood of particle j. The position of the particles are initialised randomly in the search space and the velocities are initialised to zero.

2.3. Application of the PSO algorithm to the DM problem

Each particle of the swarm implements a candidate policy. Each time the swarm moves, new candidates are considered and some exploration of the search space is performed. Yet, the computation of the fitness function for each of the particles is not possible because of the expectation (Eq. 1). Only an approximation, thanks to a Monte Carlo (MC) sampling can be computed. The MC sampling is known to be an unbiased estimator of the true function. A MC sampling consists in testing on a user the controller implemented by a particle and to compute the score for this test. Several tests can be done with one particle (so one policy) but with different users so as to estimate the fitness. For each particle j of the swarm, K tests leading to dialogues of length T_k^j are made with the policy parameterised by $_j$. The estimation is thus:

$$\hat{J}(\ _j) = \frac{1}{K}\sum_{k=1}^{K}(20 \cdot \ ^{k,j}_{\text{fulfil}} - T_k^j).$$

New candidate policies proposed during the exploration of the search space are directly presented to the users. This approach is called *on-policy* (and *online* since the data used for the optimisation are not collected beforehand). The exploration phase is necessary in order not to find sub-optimal solutions. In the usual RL framework, the exploration is usually made at the decision level: a random action is chosen from time to time. This random decision may not be consistent with what previously happened and may disturb the dialogue. Safer exploration schemes can be proposed like those presented in [21]. However, when a policy search method is used, instead of having exploration at the action level, the exploration is made on the parameters of the policy (which makes the whole policy to change).

3. Experimental settings

The test of the algorithm has been led on the tourist information task developed by the University of Cambridge. The aim for the dialogue manager is to find a venue corresponding to the user request. The request can contain up to twelve attributes. This DM uses the *Hidden Information State* (HIS) [12] paradigm to maintain a knowledge about the dialogue history and to build the current state. Notice that, even though this method is claimed to exhibit a Markovian state, no guarantees are provided. Using standard RL may thus lead to sub-optimal strategies. The DM interacts with simulated users in order for the experiments to be reproducible. The user simulation is agenda-based [25]. A goal ensures the simulator to exhibit a consistent, goal-directed behaviour. The speech understanding error rate is set to 10% by using an error simulator.

This framework has been chosen so that an accurate comparison can be made with results obtained in previous works with RL approches. Indeed, successful strategies got by using the HIS paradigm have already been obtained by means of the Gaussian Process Temporal Differences (GPTD) algorithm [26], the Least Square Policy Iteration (LSPI) algorithm [22], and the Kalman Temporal Differences (KTD) framework [22].

To determine the policy, a score function is defined according to a linear parameterisation $S_\theta(s, a) = \ ^T\Phi(s, a)$, with Φ a set of basis functions built from a Radial Basis Function (RBF) network. The policy is parameterised so as to maximise the score: $_\theta(s) = \arg\max_{a \in A} S_\theta(s, a)$. The state space in the HIS paradigm is a 4 dimensional space. The two first dimensions, s^1 and s^2 stand for the confidence score associated with the most probable dialogue histories. The third dimension s^3 is the most probable estimated user action (number of possible actions is 22). The fourth dimension s^4 is the estimation of the user goal (6 possible goals). The vector of basis functions Φ results from the concatenation of a vector of three equi-spaced Gaussians used in each continuous dimensions to tile the 2D space spanned by s^1 and s^2 with the two states s^3 and s^4. The standard deviation of the Gaussian is set to $\ = \sqrt{0.2}$. The vector Φ defined for all $(s, a) \in S \times A$ is $\Phi^T(s, a) = \ _{a,a_1}\ ^T(s), \dots, \ _{a,a_{12}}\ ^T(s)$ with $_{a,a_i}$ equal to 1 if $a = a_i$, 0 otherwise, $^T(s) = [1, \varphi_1^1(s^1, s^2), \dots, \varphi_3^3(s, s^2), s^3, s^4]$ and $\varphi_i^j = \exp(\frac{-||s^1 - s_i||^2 + ||s^2 - s_j||^2}{2\sigma^2})$, (s_i, s_j) being the RBF centers equi-spaced in $[0, 1] \times [0, 1]$. The dimension of the feature vector is thus $(1 + 9 + 2) \cdot 12 = 144$. The PSO algorithm has thus to optimise 144 parameters.

4. Results

The performance of the PSO algorithm is usually evaluated relatively to the number of calls to the fitness function computation. In Sec. 2, it is stated that at each step of the algorithm, the particles are evaluated thanks to a Monte Carlo sampling. The number of particules N_{PART} and the number of sampling N_{MC} have to be chosen by the designer of the system. For a given number of parameters and a given search space, the larger N_{PART}, the larger the exploration. Consequently, for a given number of iterations, more candidate solutions would been tested. Finding the optimal solution may be quicker in terms of iterations and thus in terms of time. Similarly, the N_{MC} number has an effect on the learning. The larger N_{MC}, the more precise yet the costlier the evaluation. Thus, several numbers of N_{PART} and N_{MC} have been tested to find a good compromise between the accuracy of the evaluation at each time step and its cost. Since the dialogue problem is stochastic, it is expected that the higher the number of samplings, the more accurate the evaluation.

The first experiments led with the DM show the behaviour of the policy returned by the best particle. The PSO algorithm is tested at different steps of the learning and for different numbers of Monte-Carlo samplings. Fig. 1 presents the test of the policy learnt by the *best particle* after N_S iterations. The number N_{PART} is set to 25. Several values of N_{MC} have been chosen to see sensibility of the method with respect to this parameter. To plot the curves, a mean over 100 PSOs is computed. For each PSO, the best policy is tested 100 times on random user goals. On the x-axis is represented the number of iterations N_s needed to perform the optimisation.

The results tend to stabilise beyond a certain Monte Carlo number of samplings ($N_{\text{MC}} = 5$). The standard deviation decreases when the number of Monte Carlo samplings increases: in this case, the approximation is more accurate and the chance of having either very good dialogues or conversely very poor dialogues is smoothered and reduced. The standard deviation also decreases when the number of steps increases with a reasonable number of N_{MC} (5 or 10): the trained policies become more efficient.

We can compare the results to those returned on the same problem by the KTD, LSPI and GPTD algorithms [22] which have been proven efficient on such task. Indeed, from the number of steps N_S, the number of samplings N_{MC} and the number of particules N_{PART}, the number of training dialogues can be deduced $N_D = N_{\text{PART}} \cdot N_{\text{MC}} \cdot N_S$. The performance are quite similar since the average number of turns in the dialogue after the learning is the same, around $20 - 15 = 5$. But the number of dialogues to reach the asymptote is larger in the PSO case (but still reasonable for a dialogue problem, around $40 \cdot 10 \cdot 7 = 2800$ versus 400 for KTD). This constraint is overtaken by the fact that the architecture is parallel. Later experiment will show that this number can be reduced. In the KTD and GPTD cases (which are on-policy algorithms) the current policy is necessarily presented to each new user calling the system at a given time. Therefore, while a dialogue is running for learning, all the users are facing the previously learnt strategy. However, in the PSO case, most of people calling at the same time see updated policies.

Moreover, one has to remind that the fitness evaluation is less informative in the PSO case than in the RL framework. In-

Figure 1: Average scores while testing the policy of the best particle ($N_{PART} = 40$, N_{MC} increases).

Figure 2: Average scores while testing the policy of the best particle. The size of the swarm changes.

deed, contrary to the RL approach where an update of the algorithm is performed at each dialogue turn thanks to an immediate reward, in the PSO approach, the evaluation is made at the dialogue level. Only the quality of a whole dialogue is used.

The number of particles also influences the learning. In the previous experiments, the influence of the number of Monte-Carlo samplings has been studied. Now, the number N_{MC} is set to 10, since in Fig. 1 this N_{MC} value returned the best results, and the number of particles N_{PART} changes. Results are presented in Fig. 2. It appears that with a size of swarm of 25 the results are still of good quality. The number of training dialogues is decreased by 1.6 in this case. Results with size of the swarm of 25 is presented in Fig. 3. After 7 iterations, the best policy seems acceptable. The number of dialogue used for the training is thus around $25 \times 10 \times 7 = 1750$.

A compromise can be found between the number of dialogues needed for the training and the quality of the policy. If the size of the swarm is set to 25 and the number of Monte-Carlo samplings is set to 10, this compromise is reached.

Figure 3: Average scores while testing the policy of the best particle ($N_{PART} = 25$, N_{MC} increases).

Figure 4: Average scores returned by the all the policies during the learning.

Fig. 4 presents the average results of the policies presented to each of the users during the learning. Each points on the graph is the result of means over all the 40 particles of each of the 100 BBOs for a different steps of the PSO. More precisely, these results correspond to the score got during all the Monte-Carlo samplings for a given iteration. Indeed, it is important to have a look at the policies presented during the learning to be ensured that too poor policies are not experimented. On average, after 20 iterations, even if the worst particles are taken into account, the policies used for training lead to successful dialogues in less than 10 steps.

5. Conclusions

This contribution proposes to use a Black Box Optimisation framework to solve a Dialogue Management problem. This approach allows some assumptions about the environment to be weakened. Indeed, usual frameworks require the Markov property to be met. Here, optimisation can be performed even if the state is not Markovian. The optimisation process is ensured to return the best reactive policy, that is the policy based on observations. In a future work, we plan to use another state for the learning than the one returned by the HIS paradigm. The potential method should just exhibit a memory to deal with the past observations and the error of the speech and semantic analysers (like a sliding window instead of a complex Bayesian framework). The BBO has also the advantage of exhibiting a parallel architecture.

The results obtained with a standard implementation of a Particle Swarm Optimisation have been compared to state of the art algorithms. The difference relies on the convergence rate. Yet, the number of data needed to find an efficient policy seems still reasonable. However, the BBO field is widely represented in the optimisation litterature. This is a prooof-of-concept paper using a standard BBO algorithm. In the future, we plan to study more efficient BBO algorithms and maybe improve the BBO litterature to fit the dialogue policy search application requirements.

6. Acknowledgements

The authors would like to thank Steve Young, Milica Gašić and the Cambridge Dialogue Systems Group for their help in using the CamInfo system. Results have been computed with the InterCell cluster funded by the Région Lorraine.

7. References

[1] S. Larsson and D. R. Traum, "Information state and dialogue management in the trindi dialogue move engine toolkit," *Natural language engineering*, vol. 6, no. 3 & 4, pp. 323–340, 2000.

[2] O. Pietquin and T. Dutoit, "Aided Design of Finite-State Dialogue Management Systems," in *Proceedings of the 4th IEEE International Conference on Multimedia and Expo (ICME)*, vol. III, Baltimore (USA, MA), July 2003, pp. 545–548.

[3] R. Freedman, "Atlas: A plan manager for mixed-initiative, multimodal dialogue," in *Proceedings of the AAAI-99 Workshop on Mixed-Initiative Intelligence*. Citeseer, 1999, pp. 1–8.

[4] R. Bellman, "A Markovian Decision Process," *Journal of Mathematics and Mechanics, vol. 6, pp. 679–684*, 1957.

[5] C. Boutilier, T. Dean, and S. Hanks, "Decision-theoretic planning: Structural assumptions and computational leverage," *Journal of Artificial Intelligence Research*, vol. 11, pp. 1–94, 1999.

[6] E. Levin, R. Pieraccini, and W. Eckert, "Learning dialogue strategies within the markov decision process framework," in *Proceedings of the workshop on the Automatic Speech Recognition and Understanding (ASRU)*. IEEE, 1997, pp. 72–79.

[7] M. Walker, D. Litman, C. Kamm, and A. Abella, "Paradise: A framework for evaluating spoken dialogue agents," in *Proceedings of Annual Meeting of the Association for Computational Linguistics*, 1997, pp. 271–280.

[8] R. Sutton and A. Barto, *Reinforcement learning: An introduction*. The MIT press, 1998.

[9] S. Singh, M. Kearns, D. Litman, and M. Walker, "Reinforcement learning for spoken dialogue systems," in *Proceedings of the Conference of the Neural Information Processing Systems Foundation (NIPS)*, 1999.

[10] E. Levin, R. Pieraccini, and W. Eckert, "A stochastic model of human-machine interaction for learning dialog strategies," *IEEE Transactions on Speech and Audio Processing*, vol. 8, no. 1, pp. 11–23, 2000.

[11] O. Pietquin and T. Dutoit, "A probabilistic framework for dialog simulation and optimal strategy learning," *IEEE Transactions on Audio, Speech and Language Processing*, vol. 14, no. 2, pp. 589–599, March 2006.

[12] S. Young, M. Gasic, S. Keizer, F. Mairesse, J. Schatzmann, B. Thomson, and K. Yu, "The hidden information state model: A practical framework for POMDP-based spoken dialogue management," *Computer Speech & Language*, vol. 24, no. 2, pp. 150–174, 2010.

[13] N. Roy, J. Pineau, and S. Thrun, "Spoken dialogue management using probabilistic reasoning," in *Proceedings of the 38th Annual Meeting on Association for Computational Linguistics*. Association for Computational Linguistics, 2000, pp. 93–100.

[14] V. Heidrich-Meisner and C. Igel, "Evolution strategies for direct policy search," *Parallel Problem Solving from Nature–PPSN X*, pp. 428–437, 2008.

[15] ——, "Similarities and differences between policy gradient methods and evolution strategies," in *Proceedings of the European Symposium on Artificial Neural Networks (ESANN)*, 2008, pp. 149–154.

[16] S. Mannor, R. Rubinstein, and Y. Gat, "The cross entropy method for fast policy search," in *Proceedings of the International Conference on Machine Learning (ICML)*, 2003, pp. 512–519.

[17] L. Busoniu, D. Ernst, B. De Schutter, and R. Babuska, "Cross-entropy optimization of control policies with adaptive basis functions," *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, vol. 41, no. 1, pp. 196–209, 2011.

[18] A. P. Engelbrecht, *Fundamentals of computational swarm intelligence*. Wiley London, 2005, vol. 1.

[19] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proceedings of the International Conference on Neural Networks (ICNN)*, vol. 4, 1995, pp. 1942–1948.

[20] M. Gašić, F. Jurčíček, S. Keizer, F. Mairesse, B. Thomson, K. Yu, and S. Young, "Gaussian processes for fast policy optimisation of POMDP-based dialogue managers," in *Proceedings of the 11th Annual Meeting of the Special Interest Group on Discourse and Dialogue (SIGdial)*, 2010, pp. 201–204.

[21] L. Daubigney, M. Gašić, S. Chandramohan, M. Geist, O. Pietquin, and S. Young, "Uncertainty management for on-line optimisation of a POMDP-based large-scale spoken dialogue system," in *Proceedings of InterSpeech*, 2011.

[22] L. Daubigney, M. Geist, S. Chandramohan, and O. Pietquin, "A Comprehensive Reinforcement Learning Framework for Dialogue Management Optimisation," *IEEE Journal of Selected Topcis in Signal Processing*, vol. 6, no. 8, pp. 891–902, 2012.

[23] J. Fix and M. Geist, "Monte-Carlo Swarm Policy Search," in *Symposium on Swarm Intelligence and Differential Evolution (SIDE)*, ser. Lecture Notes in Artificial Intelligence (LNAI). Zakopane (Poland): Springer Verlag - Heidelberg Berlin, 2012.

[24] M. Clerc, "Standard particle swarm optimisation from 2006 to 2011," *Technical Report*.

[25] J. Schatzmann, B. Thomson, K. Weilhammer, H. Ye, and S. Young, "Agenda-based user simulation for bootstrapping a pomdp dialogue system." in *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (HLT/NAACL)*, 2007, pp. 149–152.

[26] M. Gašić, F. Jurčíček, S. Keizer, F. Mairesse, B. Thomson, K. Yu, and S. Young, "Gaussian processes for fast policy optimisation of POMDP-based dialogue managers," in *Proceedings of the Annual Meeting of the Special Interest Group on Discourse and Dialogue (SIGdial)*. Association for Computational Linguistics, 2010, pp. 201–204.