Policy Optimization for Spoken Dialog Management Using Genetic Algorithm

Hang REN^{†a)}, Nonmember, Qingwei ZHAO^{†b)}, Member, and Yonghong YAN^{†,†c)}, Nonmember

SUMMARY The optimization of spoken dialog management policies is a non-trivial task due to the erroneous inputs from speech recognition and language understanding modules. The dialog manager needs to ground uncertain semantic information at times to fully understand the need of human users and successfully complete the required dialog tasks. Approaches based on reinforcement learning are currently mainstream in academia and have been proved to be effective, especially when operating in noisy environments. However, in reinforcement learning the dialog strategy is often represented by complex numeric model and thus is incomprehensible to humans. The trained policies are very difficult for dialog system designers to verify or modify, which largely limits the deployment for commercial applications. In this paper we propose a novel framework for optimizing dialog policies specified in human-readable domain language using genetic algorithm. We present learning algorithms using user simulator and real human-machine dialog corpora. Empirical experimental results show that the proposed approach can achieve competitive performance on par with some state-of-the-art reinforcement learning algorithms, while maintaining a comprehensible policy structure.

key words: spoken dialog management, spoken dialog system, genetic algorithm

1. Introduction

Spoken dialog systems (SDS) provide a natural form of human-computer interface. They have been deployed for various applications ranging from telephony information service systems to personal assistants on mobile devices. A typical configuration of SDS is shown in Fig. 1. Dialog manager (DM) receives parsed user utterances represented in semantic form and outputs system dialog action as response. DM is the central component of SDS and has a direct impact on end-to-end user experience. Its major functionalities are tracking dialog states and maintaining a dialog policy which decides how the system reacts given certain dialog state. Designing a dialog policy by hand is tedious and erroneous because of the uncertainty of underlying dialog states especially when the SDS operates in noisy environments. Thus the design of robust dialog policies has received much attention in both academia and industries. In recent years var-

Fig. 1 Diagram of a typical spoken dialog system

ious approaches for automatic DM policy optimization have been proposed [1]–[4], among which methods based on reinforcement learning (RL) and POMDP model are the most popular [5]. In RL, an agent operates in an environment and receives numeric reward as immediate feedback after conducting each action. The agent learns the optimum policy (strategy) by optimizing the cumulative reward. One of the advantages of RL-based DMs is its robustness to noisy results from automatic speech recognizer (ASR) and spoken language understanding (SLU) modules. Also, it relieves human labor involved in handcrafting DM policies by allowing the agent to automatically discover the optimal policy by exploring the underlying state-action space and incrementally improving the controlling policy.

In spite of all the advantages, RL-based DMs have not been widely employed for commercial SDSs due to several reasons [6]. Firstly, RL algorithms are often data-demanding, which leaves dialog system designers in a dilemma since there is usually little or even no data available at the early stage of system development. Several methods have been proposed to mitigate this problem. A user simulator is often firstly built using wizard-of-oz dialog data, and then the simulator is used for RL training. Approaches based on batch RL refrain from user simulation by learning policies on collected real human-machine dialog corpora. Currently methods that learn policies directly through on-line interactions with human users are also a focus in the SDS research. Secondly, in RL the learnt DM policies are often represented implicitly by the optimal value functions, which are approximated using complex numerical models. Thus the learnt policies are often beyond human comprehen-

Manuscript received February 1, 2016.

Manuscript revised April 1, 2016.

Manuscript publicized July 19, 2016.

[†]The authors are with the Key Laboratory of Speech Acoustics and Content Understanding, Institute of Acoustics, Chinese Academy of Sciences, Beijing 100190, China.

^{††}The author is with the Xinjiang Laboratory of Minority Speech and Language Information Processing, China.

a) E-mail: renhang@hcccl.ioa.ac.cn

b) E-mail: zhaoqingwei@hcccl.ioa.ac.cn

c) E-mail: yanyonghong@hcccl.ioa.ac.cn

DOI: 10.1587/transinf.2016SLP0008

sion and are difficult for system designers and maintainers to debug or modify in case that any unexpected system behaviors occur. It also keeps back domain experts from incorporating useful expert knowledge, which leaves the tuning of the policies completely to data-driven learning process.

In this paper we discuss the optimization of human-interpretable DM policies. We propose to use Genetic Algorithm (GA) [7] to optimize DM policies (GA-DM) which are comprehensible to human designers and easy to verify and modify. The underlying idea is intuitive. We use human-readable domain language to sketch the basic structure of the DM policy and leave free parameters for automatic tuning. According to previous experiences in deploying SDSs, it is relatively easy to specify a basic DM policy by hand when engineering slot-filling or task-driven SDSs of a moderate scale. The most difficult part lies in setting various threshold parameters in dealing with ASR and SLU errors via repeatedly confirming and grounding. In hand-crafted DM, these parameters are usually set heuristically or by trail-and-error. Automatic optimization of these parameters will be of great help. We hope to keep a balance between purely hand-crafted rule-based policies and the ones automatically learnt using black-box and data-driven methods while keeping the merits from both approaches. Two variants of the approach are proposed and evaluated, i.e. a direct policy searching method through interaction with a simulated user and a sample-efficient version using real human-machine dialog corpora called Q-points regression.

The remaining of this paper is organized as follows. We briefly review related work in Sect. 2 and describe the methods and algorithms in Sect. 3. In Sect. 3.1 we describe genetic algorithm and its application in DM policy optimization. We propose two kinds of fitness function based on simulation and dialog corpus in Sect. 3.2 and Sect. 3.3 respectively. In Sect. 4 we present experimental results on simulated user and real human-machine dialog corpora.

2. Related Work

Automatic DM policy optimization is a hot topic in SDS research. Many data-driven methods have been proposed among which RL-based ones are the most popular. The GASARSA [8] and KTD [9] algorithms are two of the representative methods based on on-line RL. These methods focus on high performance and sample-efficient on-line learning. However the learnt policies are mostly incomprehensible. There is some previous work on combining rule-based and RL-based DMs. In [10] Williams proposed to construct a hand-crafted DM to produce a set of candidate actions given certain dialog state, from which one action is selected by the RL-based DM. To incorporate more domain knowledge into the design of DM and constrain the behavior of the trained policies, Yoshino et al. [11] used directed intention dependency graph from rule-based DM to replace conventional probabilistic transition model in POMDP. Similarly, Lison [12] proposed to use ‘probabilistic rule’ in specifying the transition and reward models. These probabilis-

Algorithm 1 Genetic algorithm policy optimization

```

1: Input fitness function  $F$ ,  $N_{pop}$ ,  $N_{mut}$ ,  $T_{max}$ ,  $K$ ,  $\sigma$ ,  $\mu_{mut}$ 
2:  $t \leftarrow 0$ ,  $P_0 \leftarrow \emptyset$  ▷ the initial population
3: for  $i \leftarrow 1, \dots, N_{pop}$  do
4:    $P_0.add(Random.generateIndividual())$  ▷ random initialization
5:  $P_0.evalFitness()$  ▷ evaluate fitness of each individual
6: while fitness  $f_t$  not converges and  $t < T_{max}$  do
7:    $t \leftarrow t + 1$ ,  $P_t \leftarrow \emptyset$  ▷ next generation
8:    $P_t.add(P_{t-1}.getFittest())$  ▷ elitism
9:   for  $i \leftarrow 1, \dots, N_{mut}$  do
10:     $P_t.add(mutate(P_{t-1}.getFittest(), \sigma, \mu_{mut}))$  ▷ mutate the fittest
11:   for  $i \leftarrow 1, \dots, N_{pop} - N_{mut} - 1$  do
12:     $I_1, I_2 \leftarrow tournamentSelect(P_{t-1}, K)$ 
13:     $P_t.add(mutate(crossover(I_1, I_2), \sigma, \mu_{mut}))$  ▷ reproduction
14:    $P_t.evalFitness()$ 
15:    $f_t = P_t.getFittest().getFitness()$ 
16: return  $P_t.getFittest()$ 

```

```

17: function  $MUTATE(I, \sigma, \mu_{mut})$  ▷ mutate an individual  $I$ 
18:   for each parameter  $\theta_i$  of  $I$  do
19:     if  $Random.uniform() < \mu_{mut}$  then
20:        $I.\theta_i \leftarrow perturb(I.\theta_i, \sigma)$ 
21:   return  $I$ 

```

```

22: function  $PERTURB(\theta, \sigma)$  ▷ add random noise to a single parameter
23:    $g \leftarrow abs(Random.stdGaussian())$ 
24:   if  $Random.uniform() < \theta$  then
25:      $v \leftarrow -\frac{g}{\sigma} * \theta + \theta$ 
26:   else
27:      $v \leftarrow \frac{g}{\sigma} * (1.0 - \theta) + \theta$ 
28:   if  $v < 0.0$  or  $v > 1.0$  then
29:     return  $perturb(\theta, \sigma)$ 
30:   else
31:     return  $v$ 

```

```

32: function  $TOURNAMENTSELECT(P, K)$  ▷ tournament selection
33:   choose a random subset  $P_K$  of size  $K$  from  $P$ 
34:   return  $P_K.getFittest()$ 

```

```

35: function  $CROSSOVER(I_1, I_2)$  ▷ crossover of two parents
36:    $I' \leftarrow$  exchange random parts of  $I_1$  and  $I_2$ 
37:   return  $I'$ 

```

tic rules are human-readable and less parameterized. Our work shares similar intuitions to the above-mentioned methods. While the methods described in [11], [12] apply to POMDP modeling, our work focuses on decision making and does not require Bayesian state tracking. Since GA is a general optimization method, our method can be combined with a variety of dialog state trackers, including rule-based and probabilistic. In the tightly-related dialog state tracking domain, constrained Markov Bayesian polynomial is proposed [13] to embed domain knowledge. It operates in a supervised learning setting, which is rather different to the sequential decision making problem of policy learning.

The application of evolutionary methods in DM policy design is a relatively new topic. Toney et al. proposed to use extensive classifier system (XCS) for DM policies learning [14], [15]. XCS is an evolutionary reinforcement learning algorithm that is able to learn policies with more compact representation than conventional tabular RL algorithms. However the dialog state variables defined in their experiments are binary, which limits its application in real-

Listing 1 BNF grammar of dialog policy template

```

<template> ::= 'if' <cond-exp> 'then' <action> 'else' <template>
| 'if' <cond-exp> 'then' <action> 'else' <action>

<cond-exp> ::= <cond-exp> <logic-op> <cond-exp>
| <boolean-state-var>
| <num-state-var> <comparator> <free-param>

<comparator> ::= '<' | '>' | '=='

<logic-op> ::= 'and' | 'or'

```

world SDS with continuous dialog states. El Asri et al. proposed to utilize Genetic Sparse Distributed Memory to induce dialog states for value function approximation in RL. The resulted state variables are more accessible to system designers [16]. GA-based DM policy optimization was first proposed in [17]. In this paper more technical and experimental details are present and analyzed.

3. Models and Algorithms

3.1 Genetic Algorithm and Dialog Policy Template

Genetic algorithm is a general optimization framework, which simulates the evolutionary process of natural selection. In GA a *population* of candidate solutions (also called individuals or chromosomes) is kept and iteratively updated to improve the overall quality according to a specified fitness function. The population at each iteration is called a *generation*. New generation is generated according to the principle of survival of the fittest. Fitter individuals from previous generation are selected for reproduction, and then genetic operators such as crossover and mutation are used to generate new individuals. GA is a global optimization method which can solve both numerical and combinatorial problems [7]. It has been proved to be effective in solving various problems. Specifically in the field of artificial intelligence, GA has found many applications such as optimizing controlling policies in games.

In GA an individual carries all the information for a solution, which is a fixed-length real vector in our method. Each number acts as a free parameter of the dialog policy template lying in $[0, 1]$. A concrete DM policy can be instantiated by assigning the parameters of an individual to the corresponding policy template. The template is used to specify the basic structure of a dialog policy. It is defined as a set of prioritized condition-action expressions in our experiment. Given certain dialog state, condition expressions are checked sequentially in a predefined order and the first matched one is selected with the associated action chosen as output.

Listing 1 gives the BNF grammar of the proposed templates. The condition expression is parametric and parameters can be used to set thresholds for numerical state variables. In addition, parameters can also be used to induce new state variables, for example a variable representing the

Fig. 2 Crossover and mutation of individuals (chromosomes). Each individual is a real vector with constituent scalars in $[0, 1]$.

number of slots whose best scores are above certain threshold. Although the macro system action is fixed in the template, the exact semantics of each action (in this slot-filling setting, a macro action usually includes several finer-grained dialog actions with associated slots and values) can contain free parameters. For example, in the action 'offer', threshold can be used to filter hypotheses for generation of database queries.

The policy template is proposed for conciseness and simplicity but does not have to take this form exactly. The representation of a policy is intended to be easy for human interpretation and design, while parameters that adjust the DM behavior for adaptation to different environments are automatically tuned using GA. The design of the dialog template requires knowledge in the dialog domain but does not need an exact model of the operating environment. Thus it is very suitable for human experts to deal with. This engineering division is intentionally made in the proposed approach.

In Algorithm 1 our implementation of GA is described. In the main body of the GA learning a population of size N_{pop} is updated for T_{max} generations. In each generation the individuals are generated from 3 sources. The fittest individual from the previous generation is directly brought to the next generation (shown in Line 8 of Algorithm 1), and is also mutated repeatedly to generate N_{mut} additional new individuals (Line 10). This is adapted from a technique called *elitism* in the GA literature. To ensure competitive characteristics are inherited by the new generation. The rest of the population are generated by performing tournament selection and genetic operations (Line 12 and 13). For the other notations in Algorithm 1, P_t is the t -th generation and K is the tournament size used in selecting parent individuals for reproduction. μ_{mut} is the mutation rate for each parameter of the individuals and σ is a scaling factor controlling the variance of the mutated parameter.

The genetic operators *mutation* and *crossover* are shown graphically in Fig. 2. Both of them bring some randomness to the population. In crossover, random parts of the two parents are exchanged to reproduce a child individual. The mutation operator accepts a single individual as input and then scans each component sequentially, either leaving it intact or perturbing it randomly. Mutation of a single parameter is implemented by sampling from a skewed normal

distribution with the mode centered at the perturbed number. If the sampling result lies outside $[0, 1]$, the process is repeated by calling the function *perturb* recursively. This sampling sub-routine is designed to produce results with smooth probabilistic distribution. *Tournament selection* is used to select parent individuals for reproduction by randomly choosing K individuals from the population and returning the fittest one. In this process fitter individuals are more likely to be chosen while it is also possible for the less fit ones to be selected. These methods are used to search the underlying solution space effectively, other useful GA techniques can also be utilized and this is an interesting topic for future study.

The fitness function F is essential to the success of GA since it acts as the optimization objective for GA. Two kinds of fitness functions are proposed in the following sections to evaluate DM policies by using user simulators and dialog corpora respectively.

3.2 DM Policy Optimization Using User Simulation

User simulation is a commonly adopted approach in dialog research since interacting with real user is often time-consuming and labor-intensive. With a user simulator available, the performance of a DM policy can be directly evaluated using Monte Carlo experiments. An agenda-based user simulator is utilized [18] and N dialog sessions are conducted between the simulated user and the DM to be evaluated. Average cumulative reward is calculated as the fitness for the individual, which is the same to the objective of RL algorithms,

$$F_R[\pi_{GA}] = \frac{1}{N} \sum_{i=1}^N \sum_{t=1}^{T_i-1} \gamma^{t-1} r_{i,t}, \quad (1)$$

where i and t are indices for dialog session and dialog turn respectively, T_i is the number of dialog turns of session i , $r_{i,t}$ is the immediate reward received at a turn and γ is the discounting rate. In the following sections we use similar notations.

A noise channel is designed to simulate ASR and SLU errors. Given uncorrupted SLU results, confident scores are randomly generated and then noise-polluted results are generated according to the scores. For each dialog act represented as $\{\text{act}, (\text{slot}, \text{value})\}$, replacement and deletion are randomly applied to *value*. The polluted results are aggregated to form N -best hypotheses which are then fed into DMs.

3.3 Q-Points Regression on Dialog Corpora

It is a non-trivial task to build a user simulator that mimics human behavior exactly, and it is also difficult to measure the inconsistency between a simulated user and a real one. One of the solutions proposed is to use batch RL [19] or hybrid learning [20] to learn DM policies directly on dialog

Algorithm 2 Episodic fitted Q-iteration

```

1: Input  $\{(s_{i,t}, a_{i,t}, s_{i,t+1}) | i \leftarrow 1, \dots, N, t \leftarrow 1, \dots, T_i - 1\}$ 
2: initialize Q-function approximator  $\hat{Q}(s, a)$  and array  $\{Q_{i,t}\}$ 
3: for  $l \leftarrow 1, \dots, L_{max}$  do ▷ run for  $L_{max}$  iterations
4:   for  $i \leftarrow 1, \dots, N$  do ▷ enumerate  $N$  dialog sessions
5:     for  $t \leftarrow 1, \dots, T_i - 1$  do ▷ enumerate the  $T_i$  turns of dialog  $i$ 
6:        $r_{i,t} \leftarrow \text{reward}(s_{i,t}, a_{i,t}, s_{i,t+1})$ 
7:       if  $t == T_i - 1$  then
8:          $Q_{i,t} \leftarrow r_{i,t}$  ▷ when the dialog ends
9:       else
10:         $Q_{i,t} \leftarrow r_{i,t} + \gamma \max_a \hat{Q}(s_{i,t+1}, a)$ 
11:      update  $\hat{Q}(s, a)$  by performing supervised learning on  $\{(s_{i,t}, a_{i,t}), Q_{i,t}\}$ 
12: return  $\hat{Q}(s, a)$ 

```

corpora. We propose a fitness function to support batch-style policy learning using GA. First, a batch RL algorithm is performed on the corpus, inducing an estimated optimum Q-function $\hat{Q}(s, a)$, and an corresponding implicitly defined policy $\pi_Q(s) = \arg \max_a \hat{Q}(s, a)$. Then the learnt value function $\hat{Q}(s, a)$ is used in the definition of the fitness function. Here s and a denote dialog state and dialog action respectively. Fitted Q-iteration (FQI) [21] is utilized for batch RL and is described in Algorithm 2. FQI has been applied to DM policies optimization and shows high data-efficiency [19]. The inputs to the algorithm are state-action-next-state triplets in the form of $\{(s_{i,t}, a_{i,t}, s_{i,t+1})\}$. Sample-based Bellman backup operator (Line 10) is used to update the estimated Q-values, and then supervised learning is performed on the training samples to update the approximate value function. This process is repeated for the convergence of $\hat{Q}(s, a)$. Extremely Random Trees (ExtraTrees) [22] are used for value function approximation. ExtraTrees are a powerful model for regression and classification. They can learn non-linear patterns in the inputs and are less susceptible to over-fitting.

Two fitness functions are proposed based on different heuristics. Given a DM policy π_{GA} , the *NPoints* fitness function calculates the number of dialog turns where the actions predicted by π_{GA} and π_Q are identical.

$$F_{NPoints}[\pi_{GA}] = \sum_{i=1}^N \sum_{t=1}^{T_i-1} \delta(\pi_{GA}(s_{i,t}), \pi_Q(s_{i,t})), \quad (2)$$

where $\delta(a_1, a_2) = 1$ only when $a_1 = a_2$ and $\delta(a_1, a_2) = 0$ otherwise. This fitness calculation does not evaluate the relative performance gap of each action. To take this piece of information into account, we also propose another form of fitness function called *QVal*, which calculates the sum of values for the actions predicted by π_{GA} on the training samples. The value function trained on a fixed corpus is often inaccurate in less-explored regions of the state-action space [20], [23]. To mitigate the problem a supervised classifier $\hat{P}(a|s)$ is built on the training set with the actual observed actions as targets. If the probability for an action is greater than a predefined threshold λ , the value $\hat{Q}(s, a)$ is used, otherwise a constant R is used for punishment.

$$F_{QVal}[\pi_{GA}] = \sum_{i=1}^N \sum_{t=1}^{T_i-1} \tilde{Q}_\lambda(s_{i,t}, \pi_{GA}(s_{i,t})) \quad (3)$$

$$\tilde{Q}_\lambda(s, a) = \begin{cases} \hat{Q}(s, a) & \text{if } \hat{P}(a|s) > \lambda \\ R & \text{otherwise.} \end{cases}$$

These two fitness functions obviously differ in weighing the importance of training instances. F_{QVal} puts a greater effort in optimizing against instances with larger Q-value gaps between different actions and avoids to take less likely actions. Combining GA with the above two methods leads to the Q-points regression algorithm. Note that one limitation of this algorithm compared to learning using a simulator is that the semantics of the macro actions should be fixed and thus no free parameters can be used to adjust the action structure. The fitness is estimated on the results of batch RL, which does not support dynamic change of action structure.

3.4 DM Policy Evaluation Using Dialog Corpora

We describe a DM policy performance evaluation method on dialog corpus to avoid the need for deploying the DM on-line. We estimate a policy by learning its value function using a similar iterative routine to FQI on a held-out testing dialog corpus. The estimated cumulative reward when following the policy is used as a metric for performance. A similar approach has been taken in evaluating the effect of dialog state tracker on end-to-end performance of SDS [23]. To estimate the Q-function of a DM policy, the Bellman backup operation (Line 10) in Algorithm 2 is changed to:

$$Q_{i,t} \leftarrow r_{i,t} + \gamma \hat{Q}(s_{i,t+1}, \pi(s_{i,t+1})), \quad (4)$$

where π is the DM policy to evaluate. After the value function converges, the average reward for all the training samples of initial dialog turn $\frac{1}{N} \sum_i Q_{i,0}$ is used as a metric for performance.

4. Empirical Evaluation Results

In this section, the proposed methods are empirically evaluated based on simulated user and dialog corpora.

4.1 Experiment Settings

We devise a restaurant information domain for dialog simulation. There are 4 slots for the user simulator to fill before a database query. During simulations the DM interacts with the user simulator with a noise channel in between. The noise level of the channel can be adjusted to simulate different environmental noise conditions. Since the simulation process is stochastic, each experiment is conducted for 100 times, and the mean and standard deviation of testing performance are reported.

The reward function for the simulated environment is defined as follows. At each dialog turn the agent receives -1.0 reward. If correct restaurants are offered to users, 100.0 points are rewarded. But if the information is duplicate to

that previously offered or the presented restaurants do not match user goal, -5.0 points are given. The reward discounting rate γ is set to 0.9.

In the evaluation on dialog corpora, the DSTC2 public dataset [24] is used for both DM policy learning and evaluation. We use the original DSTC2 testing set which consists of 1117 dialog sessions. These dialog sessions are randomly and (almost) equally split into new training and testing data sets for our experiments. The DSTC2 dataset was originally designed as a benchmark corpus for evaluating the performance of dialog state tracking algorithms. With annotations for dialog states, actions, SLU outputs and other information, it has also been used to evaluate end-to-end DM performance [23].

The dialog states used in both the simulated and on-corpus experiments include both continuous and discrete variables. The state variables comprise confidence scores for each slot considering the whole dialog history. User and system dialog actions in the current turn are also included.

4.2 Experiments on Simulated User

The dialog policy template used in the simulated experiments is shown as follows.

- c0** On dialog beginning: **Welcome**
- c1** There are no valid SLU results or the top SLU hypothesis score is less than θ_0 : **Repeat**
- c2** User has just denied a slot: **Request** that slot
- c3** There is a slot with score less than θ_1 in the tracker:
if the score is larger than θ_2 then **ExplicitConf** else **Request**
- c4** The system has not yet output the action **RequireMore**: **RequireMore**
- c5** Otherwise: query the database with slot-value pairs whose scores are greater than θ_3

The template has 6 condition-action clauses and contains 4 free parameters. Note that 3 free parameters lie in the condition expressions while θ_3 is used to adapt the semantics of the macro action **offer**, which queries the database and presents results to the user.

A rule-based DM policy is built by setting the 4 parameters heuristically. A RL-based policy trained using Q-learning with linear approximation is also built for comparison with the proposed methods. It is trained for over 100,000 dialog sessions to ensure that the state-action space is sufficiently explored and the optimal performance is reached. The probability for exploration is set to 0.3. In the training process of each kind of DM, the noise level is increased incrementally to encourage the DMs to optimize the overall performance for a series of noise conditions. The same reward function and discounting rate are also used in the fitness estimation of GA. We run GA for 30 generations in policy training.

During testing 1000 dialog sessions are conducted and the noise level is adjusted in the same way as in training.

Fig. 3 Overall testing performance of DM policies. (a) Performance of RL-based, rule-based and GA-DM policy. (b) Performance of GA-DM with one clause disabled, where ‘-C1’ means that C1 is disabled.

We report both the overall performance (average reward received under a series of noise conditions) and the performance with a fixed noise level. The overall testing performance of each DM is shown in Fig. 3. Performance when operating under fixed noise condition is shown in Fig. 4 and 5. The level of environmental noise is measured using the semantic error rate of the top hypothesis of SLU results. It should be emphasized that the noise levels shown in the results are the same ones used in training. In addition to the GA-DM using the complete policy template, the utility of each individual clause in the template is evaluated. The four major clauses c1-c4 are disabled sequentially. The resulted DMs are evaluated using the same settings, and the testing results are shown along with the full-fledged GA-DM. The effects of different GA population size are explored and reported in Fig. 6.

Since the DMs are optimized against the average reward received under several noise conditions, the overall testing reward shown in Fig. 3 should be taken as the direct metric of performance. The RL-based policy showed better overall performance than the rule-based one, while GA-DM significantly outperformed both the rule-based and RL-based policies. From Fig. 4, it can be seen that when the noise is low, the rule-based DM is very competitive and shows even better performance than the RL-based DM. But when the noise level of the environment increases, its performance degrades seriously, while the RL-based DM is much more robust. However, after tuning of the free parameters using GA, the GA-DM outperforms both the other DMs on nearly all noise conditions. Note the maximum noise level at which each DM could successfully complete a dialog, suggesting that the GA-DM is able to operate under

Fig. 4 On-line evaluation of GA-DM using a simulated user with fixed noise level. Average cumulative reward, dialog length and task completion rate are plotted against the error rate of the top SLU hypothesis, which is used as a metric for environmental noise.

more adverse environment. It is worth mentioning again that the rule-based DM and GA-DM are instantiations of the same policy template. The simulation results justify GA as an effective method for DM policy optimization and reveal the performance potential of simple and yet human-interpretable DM policies.

It is interesting to make a comparison between RL and GA policy learning. In DM policy optimization, the state space is often continuous and infinite. In conventional RL, a model of the underlying optimal value function of the environment has to be designated. The ability of the model to approximate the optimal value function is a key factor affecting the performance of the learnt policy. However, the design of the model is often non-intuitive and complicated since it operates in the value function space. Expert knowledge is often difficult to be directly applied. This fact can help to explain that in our experiments, the RL-based DM is not as competitive as the others when the noise level is low. Since the noise level is varied during training, the resulted learning environment is much more difficult to deal with than one with fixed noise condition. Thus the linear model used is unlikely to perfectly match the underlying optimal value function and cannot accommodate all types of condition. In our experiment the RL policy has learnt to make a trade-off and adapted to conditions with high environmental noise for a better overall performance. GA-DM tackles the problem from a different perspective. It operates in policy space directly and is much easier to incorporate expert knowledge. In GA-DM a policy model is developed

Fig. 5 Performance of the model operating with fixed noise level when the major clauses c1-c4 are sequentially disabled and retrained. Full GA-DM is the original model with all clauses enabled.

instead. Equivalent assumptions about policy structure are often difficult to made in value function space. Thus the resulted policy model can be more powerful and expressive than one for value function.

The relative utility of each clause of the policy template on the performance is another interesting aspect to be investigated. According to the results shown in Fig. 3 (b) and Fig. 5, it can be observed that when C2 is disabled the performance drops seriously. But to our surprise, when C4 is disabled, the performance significantly boosts especially in high-noise regions. The results show the relative utility of each clause in the template and reveal the necessity to optimize the structure of policy template. This kind of structural optimization problem can also be solved using GA, and we plan to study this kind of optimization in future work.

In GA the population size often influences the optimization efficiency. The training fitness and testing performance using different population size is shown in Fig. 6. We can observe that with an increasing population size, the training and testing performance nearly monotonically increases. This performance improvements are more obvious when the size is less than 100, and are not noticeable above 300. Because the elitism technique is used and the fitness of the elitist individual is cached, the training fitness improves

Fig. 6 Training fitness and testing performance of GA-DM when trained using different population size. Standard deviations are plotted as shaded areas and error bars.

steadily during training.

From the aspect of computation complexity, in GA-DM training, most computation is needed for the evaluation of fitness. Assuming that the fitness estimation for one individual takes $O(T_f)$, the maximum number of generations is set to T_{max} and the population size is N_{pop} , the total computation for fitness estimation takes $O(T_{max}N_{pop}T_f)$. This computation is somewhat heavy, but luckily the fitness estimation for the N_{pop} individuals in the same generation can be conducted concurrently. So the computation time can be greatly reduced through a parallel implementation.

4.3 Experiments on Dialog Corpora

The testing set of the DSTC2 corpus is used for on-corpus DM learning and evaluation [24]. We use the results produced by the ‘focus’ state tracker as dialog state input using the scripts provided by the DSTC2 organizers. The dialog template used in GA-DM comprises 9 condition-action clauses and 6 free parameters.

In addition to the GA-DMs with two different fitness functions as described in Sect. 3.3, the results of 3 additional DMs are shown for comparison.

1. **SL-Original** which is trained using supervised learning with the original dialog actions as target using the

Table 1 Estimated cumulative reward of DM policies on training and testing set. Numbers in brackets are standard deviations estimated by re-sampling experiments.

DM	Training	Testing
GA-NPoints	98.46 (38.30)	89.52 (41.30)
GA-QVal	127.38 (5.59)	129.29 (7.90)
SL-Original	115.63 (4.08)	114.39 (6.07)
SL-MaxQ	245.19 (12.59)	53.46 (36.06)
ThresholdedQ	142.48 (4.22)	122.21 (4.36)

ExtraTrees classifier. The probability produced by the classifier is represented as $\hat{P}(a|s)$ and it is also used for later processing.

2. **SL-MaxQ** which is also trained in a supervised manner using the actions with the maximum Q-value predicted by $\hat{Q}(s, a)$ as learning targets.
3. **ThresholdedQ** as is described in [23]. It selects the action with the maximum Q-value predicted by $\hat{Q}(s, a)$ from the set of actions whose probabilities predicted by $\hat{P}(a|s)$ are greater than λ . This thresholding technique is used to constrain the behavior of RL policy, in case of insufficient exploration of the state-action space.

To obtain more stable estimation of the performance, the experiment is repeated for 12 times and in each one the original dataset is reshuffled and split to get new training and testing instances. The policy evaluation method described in Sect. 3.4 is used on the testing sets. The mean and standard deviation of the estimated reward for the initial dialog turns are reported in Table 1.

The **SL-MaxQ** which acts greedily upon $\hat{Q}(s, a)$ has poor performance on the testing set while being overrated on the training set. In [20] Henderson et al. showed that policies learnt using RL on fixed dataset can exhibit irregular behavior when deployed on-line due to insufficient exploration of state-action space, and they proposed to use a supervised learner to mitigate the problem. We speculate that this performance degradation of **SL-MaxQ** can be attributed to the insufficient exploration problem. The **ThresholdedQ** mitigates the problem significantly by using a supervised learner and setting a simple threshold. That heuristic is shared with the **QVal** fitness function. **GA-QVal** outperforms all the other DMs and is very stable across the re-sampling experiments considering the relatively low deviation, while the behavior of **GA-NPoints** is less consistent resulting in an overall bad performance. Although **GA-QVal** is trained under the guidance of RL, its performance is superior to both **SL-MaxQ** and **ThresholdedQ**, which should be attributed to the prior domain knowledge incorporated into the policy template. The DMs in boldface outperform **SL-Original** which is built by imitating the policy used for producing the dialog corpus. This indicates the possibility of building a better and yet human-comprehensible DM policy using existing dialog corpus. A notable advantage of the Q-points regression is that this approach is much more sample-efficient, while the fitness evaluation using Monte Carlo experiments is only possible using simulation.

5. Conclusions and Future Work

In this paper a dialog policy optimization framework based on genetic algorithm is proposed. Two kinds of fitness evaluation routines are proposed, which are based on interacting with a simulated user and using dialog corpora respectively. Experiments are conducted in simulated environment and using the DSTC2 corpus. Favorable results for the proposed methods are shown. From the results we can conclude that by setting up appropriate free parameters, the performance of simple rule-based DM policies can be largely improved, and can even outperform those trained using advanced RL algorithms. A great advantage of the proposed framework is that the trained policy is easy for human to interpret and verify, which makes the method potentially more suitable for commercial application. It is also possible to upgrade existing rule-based DMs to a parametric form and using collected data to optimize the free parameters.

This paper represents a preliminary attempt to apply evolutionary methods to SDS research. Many aspects need further study, especially the design of fitness functions. Experimental results reveal the need to automatically optimize the dialog policy structure. We are interested in the structural optimization using GA and plan to study extensions of the proposed approach in future work.

Acknowledgments

This work is partially supported by the National Natural Science Foundation of China (Nos. 11461141004, 61271426, 11504406, 11590770, 11590771, 11590772, 11590773, 11590774), the Strategic Priority Research Program of the Chinese Academy of Sciences (Nos. XDA06030100, XDA06030500, XDA06040603), National 863 Program (No. 2015AA016306), National 973 Program (No. 2013CB329302) and the Key Science and Technology Project of the Xinjiang Uygur Autonomous Region (No. 201230118-3).

References

- [1] J.D. Williams and S. Young, "Partially observable Markov decision processes for spoken dialog systems," *Computer Speech & Language*, vol.21, no.2, pp.393–422, 2007.
- [2] S. Young, M. Gašić, 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.
- [3] C. Lee, S. Jung, S. Kim, and G.G. Lee, "Example-based dialog modeling for practical multi-domain dialog system," *Speech Communication*, vol.51, no.5, pp.466–484, 2009.
- [4] P. Lison, *Structured Probabilistic Modelling for Dialogue Management*, Ph.D. thesis, University of Oslo, 2014.
- [5] S. Young, M. Gašić, B. Thomson, and J.D. Williams, "POMDP-Based Statistical Spoken Dialog Systems: A Review," *Proceedings of the IEEE*, vol.101, no.5, pp.1160–1179, 2013.
- [6] T. Paek, "Reinforcement learning for spoken dialogue systems: Comparing strengths and weaknesses for practical deployment," *Proceedings of Dialog-on-Dialog Workshop, Interspeech*, 2006.

- [7] D. Whitley, "A genetic algorithm tutorial," *Statistics and Computing*, vol.4, no.2, pp.65–85, June 1994.
- [8] M. Gasic and S. Young, "Gaussian Processes for POMDP-Based Dialogue Manager Optimization," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol.22, no.1, pp.28–40, 2014.
- [9] L. Daubigny, M. Geist, S. ChandraMohan, and O. Pietquin, "A Comprehensive Reinforcement Learning Framework for Dialogue Management Optimization," *IEEE Journal of Selected Topics in Signal Processing*, vol.6, no.8, pp.891–902, 2012.
- [10] J.D. Williams, "The best of both worlds: unifying conventional dialog systems and POMDPs," *INTERSPEECH*, pp.1173–1176, 2008.
- [11] K. Yoshino, S. Watanabe, J. Le Roux, and J.R. Hershey, "Statistical dialogue management using intention dependency graph," *IJCNLP*, pp.962–966, 2013.
- [12] P. Lison, "A hybrid approach to dialogue management based on probabilistic rules," *Computer Speech & Language*, vol.34, no.1, pp.232–255, 2015.
- [13] K. Yu, K. Sun, L. Chen, and S. Zhu, "Constrained markov bayesian polynomial for efficient dialogue state tracking," *Audio, Speech, and Language Processing, IEEE/ACM Transactions on*, vol.23, no.12, pp.2177–2188, 2015.
- [14] D. Toney, J. Moore, and O. Lemon, "Developing conversational interfaces with XCS," *9th International Workshop on Learning Classifier Systems (IWLCS)*, 2006.
- [15] D. Toney, J. Moore, and O. Lemon, "Evolving optimal inspectable strategies for spoken dialogue systems," *Proceedings of the Human Language Technology Conference of the NAACL, New York City, USA*, pp.173–176, Association for Computational Linguistics, June 2006.
- [16] L.E. Asri, R. Laroche, and O. Pietquin, "Compact and interpretable dialogue state representation with genetic sparse distributed memory," *Proceedings of the 7th International Workshop on Spoken Dialogue Systems (IWSDS 2016)*, Saariselka, Finland, Jan. 2016.
- [17] H. Ren, W. Xu, and Y. Yan, "Optimizing human-interpretable dialog management policy using genetic algorithm," *Proceedings of the Automatic Speech Recognition and Understanding Workshop (ASRU 2015)*, Scottsdale, USA, pp.791–797, Dec. 2015.
- [18] J. Schatzmann, B. Thomson, K. Weilhammer, H. Ye, and S. Young, "Agenda-based user simulation for bootstrapping a POMDP dialogue system," *Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Companion Volume, Short Papers*, pp.149–152, Association for Computational Linguistics, 2007.
- [19] O. Pietquin, M. Geist, S. Chandramohan, and H. Frezza-Buet, "Sample-efficient batch reinforcement learning for dialogue management optimization," *ACM Transactions on Speech and Language Processing (TSLP)*, vol.7, no.3, pp.1–21, 2011.
- [20] J. Henderson, O. Lemon, and K. Georgila, "Hybrid Reinforcement/Supervised Learning of Dialogue Policies from Fixed Data Sets," *Computational Linguistics*, vol.34, no.4, pp.487–511, 2008.
- [21] D. Ernst, P. Geurts, and L. Wehenkel, "Tree-based batch mode reinforcement learning," *J. Machine Learning Research*, pp.503–556, 2005.
- [22] P. Geurts, D. Ernst, and L. Wehenkel, "Extremely randomized trees," *Machine Learning*, vol.63, no.1, pp.3–42, March 2006.
- [23] S. Lee, "Extrinsic Evaluation of Dialog State Tracking and Predictive Metrics for Dialog Policy Optimization," *15th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pp.310–317, 2014.
- [24] M. Henderson, B. Thomson, and J.D. Williams, "The second dialog state tracking challenge," *15th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pp.263–272, 2014.

Hang Ren received his B.Eng. in electronic and information engineering from Harbin Institute of Technology in 2007. Currently he is a Ph.D. candidate at Institute of Acoustics, Chinese Academy of Sciences. His research interests include spoken dialog system and nature language processing.

Qingwei Zhao received his Ph.D. in electronic engineering from Tsinghua University in 1999. He is a professor at Institute of Acoustics, Chinese Academy of Sciences. His research interests include spontaneous speech recognition, keyword spotting and automatic pronunciation evaluation.

Yonghong Yan received his B.Eng. from Tsinghua University in 1990, and his Ph.D. from Oregon Graduate Institute (OGI). He worked at OGI as an Assistant Professor (1995), Associate Professor (1998) and Associate Director (1997). From 1998 to 2001 he worked at Intel as Principal Engineer of Microprocessor Research Lab and Director of Intel China Research Center. Currently he is a professor and director of the Key Laboratory of Speech Acoustics and Content Understanding at Institute of Acoustics, Chinese Academy of Sciences. His research interests include speech processing and recognition, language/speaker recognition, and human computer interface. He has published more than 100 papers and holds 40 patents.