Robust Learning in Expert Networks: A Comparative Analysis

Ashiqur R. KhudaBukhsh, Jaime G. Carbonell, and Peter J. Jansen

Carnegie Mellon University
{akhudabu, jgc, pjj}@cs.cmu.edu

Abstract Learning how to refer effectively in an expert-referral network is an emerging challenge at the intersection of Active Learning and Multi-Agent Reinforcement Learning. Distributed interval estimation learning (DIEL) was previously found to be promising for learning appropriate referral choices, compared to greedy and Q-learning methods. This paper extends these results in several directions: First, learning methods with several multi-armed bandit (MAB) algorithms are compared along with greedy variants, each optimized individually. Second, DIEL's rapid performance gain in the early phase of learning proved equally convincing in the case of multi-hop referral, a condition not heretofore explored. Third, a robustness analysis across the learning algorithms, with an emphasis on capacity constraints and evolving networks (experts dropping out and new experts of unknown performance entering) shows rapid recovery. Fourth, the referral paradigm is successfully extended to teams of Stochastic Local Search (SLS) SAT solvers with different capabilities.

Keywords: active learning; referral networks; SLS SAT solvers

1 Introduction

Human experts may refer to other experts when given problems outside their area of expertise. In a clinical network, a physician may diagnose and treat a patient or refer the patient to another physician whom she believes may have more appropriate knowledge, conditioned on the presenting symptoms. Referral networks are common across other professions as well, such as members of large consultancy firms. In addition to human professional networks, we can envision referral networks of automated agents, such as the SAT solver network introduced in this paper, or heterogeneous networks of human and machine agents. In all cases, unless the experts are omniscient they often need to consult or refer problems to their colleagues. This paper addresses learning to whom an expert should refer given a problem and topic, attempting to optimize performance based on accumulated experience of the results of previous referrals. We address the distributed learning-to-refer setting, without a "boss agent" telling all the others when to try and solve a problem or when to refer and if so to whom.

Much of this work is based on DIEL, a simple but efficient algorithm balancing exploration with exploitation, which has been proposed in [1]. Previous work is summarized in Section 2. Section 3 presents our assumptions and the structure

of referral networks and expertise. Sections 4, 5 and 6 describe the distributed learning algorithms we used in the comparison, our experimental setup, and the results.

2 Related Work

Our primary predecessor for this work was the referral learning framework proposed in [1], although that work did not extend to comparison among competing algorithms, nor address capacity constraints, nor robustness to unexpected dropouts and additions to the network, which is needed for modeling actual networks of human experts. This paper also addresses fully-autonomous SAT-solver agents, vs just simulated experts. A completely different approach was taken in [2], which extends the same referral framework by augmenting the learning setting with an advertising mechanism, where experts can post estimates of their skill level in different tasks, encouraging truth-in-advertisement.

The problem of learning appropriate referrals can be cast in various ways. Taking it as a multi-armed bandit (MAB) selection problem, we accordingly enlisted several MAB algorithms [3, 4, 5] with known finite-time regret bound for performance comparison, none of which have as far as we know been studied in the context of referral learning before. An analysis of referral networks also exhibits similarities with the study of task allocation [6, 7], where minimizing turn-around time corresponds to the maximizing the probability of a correct answer. The `FAL` algorithm described in [7] uses a variant of ϵ-greedy Q-learning similar to one we compared favorably against in the current work.

Finally, in part of this work, we used `SATenstein` [8], a highly parameterized Stochastic Local Search (SLS) SAT solver to generate non-synthetic experts and expertise data. This was done previously [9] in the context of the augmented setting discussed in [2]. Here we systematically apply a set of referral algorithms (as opposed to two), and use continuous rewards (as opposed to binary) to introduce the notion of *solution quality*.

3 Referral Networks

A *referral network* can be represented by a graph (V, E) of size k in which each vertex v_i corresponds to an expert e_i $(1 \leq k)$ and each bidirectional edge $\langle v_i, v_j \rangle$ indicates a *referral link*. We call the set of experts linked to an expert e_i by a referral link, the *subnetwork* of expert e_i. In a referral *scenario*, a set of m instances $(q_1, \ldots, q_m)$ belonging to n topics $(t_1, \ldots, t_n)$ are to be addressed by the k experts $(e_1, \ldots, e_k)$.

Assuming a query budget of $Q = 2$, the following steps are executed for each instance q_j.

1. A user issues an *initial query* to an expert e_i (*initial expert*) chosen uniformly at random.
2. Expert e_i examines the instance and solves it if able, depending on the *expertise* of e_i wrt. q_j, defined as the probability that e_i can solve q_j correctly.
3. If not, she passes a *referral query* to a *referred expert* within her subnetwork. *Learning-to-refer* means improving the estimate of who is most likely to solve the problem.

4. If the referred expert succeeds, she communicates the solution to the initial expert, who in turn, communicates it to the user.

We also considered in our experiments the case $Q > 2$, when the recipient of a referral can herself re-refer to another expert (with reduced budget).

For the simulations, we follow (initially) the *assumptions* made in [1], notably that: network connectivity depends on (cosine) similarity between the topical expertise, expertise is stationary, and its distribution can be characterized by a mixture of Gaussian distributions (for further details, see [1, 2]).

4 Distributed Referral Learning

Essentially, from the point of view of a single expert, learning appropriate referral choices for a given topic is an action selection problem. Action selection using Interval Estimation Learning (IEL) works in the following way [2, 10]. First, for each action a, the upper confidence interval for the mean reward ($UI(a)$) is estimated by

$$UI(a) = m(a) + \frac{s(a)}{\sqrt{n}} \tag{1}$$

where $m(a)$ is the mean observed reward for a, $s(a)$ is the sample standard deviation of the reward, n is the number of observed samples from a. Next, IEL selects the action with the highest upper confidence interval. The intuition behind selecting the action with the highest mean plus upper confidence interval is that high mean indicates good expected performance and high variance indicates we lack knowledge about said performance. This naturally trades off exploitation (selecting high mean) and exploration (resulting in variance reduction). An earlier version of DIEL [1] used an additional Student's t-distribution parameter, here we are using the version reported in [2], which is parameterless and outperformed the earlier version.

In a distributed setting, each expert is running a thread of action selection for each topic in parallel. So basically, DIEL consists of multiple IELs for each topic/expert pair. Categorized into three broad categories: Q-Learning variants, UCB-variants and greedy-variants, our choice of referral algorithms is presented in Table 1. The distributed versions of these algorithms function the same way as DIEL – only their action selection procedure is different. Ideally, the distributed version of UCB1 should be called DUCB1; but since we have both Q-Learning [11] and DQ-Learning [12] in our pool of algorithms, we slightly abuse the notation in order to avoid confusion.

A primary challenge in the distributed setting is that there is no global visibility of rewards, i.e., $reward(e_i, topic_p, e_j)$ (a function of the initial expert e_i, instance topic $topic_p$, and referred expert e_j) is only visible to expert e_i. Also, because of the scale, in a practical setting, we cannot afford a large number of referrals for finding suitable referral choices. For this reason, a high performance in the early phase of learning is crucial.

In the following description of the action selection procedures, $m(a)$ is the mean observed reward for action a, n_a is the number of observations of a, and N

Table 1. Referral Algorithms

Category	Algorithm	Parameters
IEL	DIEL [2]	None
Greedy	DMT [1]	None
Greedy	ϵ-Greedy [13]	c
Greedy	ϵ-Greedy1	α
UCB	UCB1 [3]	None
UCB	UCB2 [13]	None
UCB	UCBNormal [4]	None
UCB	UCBV [5]	θ
Q-learning	Q-learning [11]	α, γ, ϵ
Q-learning	DQ-learning [12]	α, γ, ϵ

is the total number for all actions.

DMT: Unlike DIEL, DMT only considers the mean observed reward and always greedily picks the action with the highest mean reward.

ϵ-Greedy: DMT, being purely greedy, can easily get stuck with a sub-optimal referral choice. **ϵ-Greedy** performs a diversification step with a probability ϵ. i.e., with probability ϵ, it randomly chooses one of the connected experts for referral.

ϵ-Greedy1: ϵ-Greedy1 differs from **ϵ-Greedy** only in its way of setting the diversification probability parameter (set to $\frac{\alpha * K}{N}$ where K is the subnetwork size, i.e., the total number of referral choices).

UCB1: UCB1 selects the action with highest $m(a) + \sqrt{\frac{2lnN}{n_a}}$. This implies among two actions with equal mean reward, UCB1 will favor the least sampled one.

UCB2: UCB2 executes in an episodic fashion. Once an action is selected, it is executed for an episode. For each action a, it first initializes r_a to 0 where r_a denotes the episode length and each action is executed once in the beginning. If the last selected action j has been played for r_j times in a row, the new action is selected by maximizing $m(a) + \sqrt{\frac{(1+\alpha).ln(eN\tau(r_a))}{2\tau(r_a)}}$ where $\tau(r_a) = (1+\alpha)^{r_a}$ and α is a configurable parameter.

UCB-normal: UCB-normal performs any action that has been executed less than $\lceil logN \rceil$. Otherwise, the action with highest $m(a) + \sqrt{16.\frac{sq(a) - n_a.m(a)^2}{n_a - 1}.\frac{ln(N-1)}{n_a}}$ is chosen ($sq(a)$ is the sum of squared rewards obtained from action a).

UCBV: Similar to DIEL, UCBV also uses variance to compute expected reward. However, it uses a different exploration function, $\frac{logN}{n_a}$. UCBV selects the action with highest $m(a) + s(a).\sqrt{\frac{2\theta log(N)}{n_a}} + \frac{3\theta log(N)}{n_a}$. [5] reported a value of 1.2 for the parameter θ to guarantee logarithmic convergence.

DQ-Learning: Double Q-Learning, or DQ-learning, consists of two standard Q-Learning algorithms running in tandem (with Q functions: Q_A and Q_B, say). Whenever an action is chosen based on Q_A, the observed reward is used to update Q_B and vice versa. In practice, DQ-Learning tends to converge faster than Q-Learning (for further details, see [12]). For Q-Learning, we considered ϵ-Greedy-Q-Learning.

5 Experimental Setup

We compared the DIEL referral-learning algorithm against the nine other algorithms in Table 1), a topical upper bound, and a random (expertise-blind) baseline. Each parameterized referral algorithm was tuned on a separate training set constructed using the same parameter distribution described in [1]. For each algorithm, we ran 100 random instantiations of the algorithm on the training data set and selected the configuration that performed best on this set. The ϵ-Greedy algorithm, as presented in [13], requires prior knowledge about the reward distribution in order to set the value of the hyper-parameter d. However, we found that estimating d from the observations created sub-par performance. Setting instead ϵ to $\frac{\alpha * K}{N}$ (where K is the subnetwork size and N is the number of total observations) gave rise to a good performance when appropriately configured. We followed a similar procedure to set ϵ for ϵ-Greedy Q-learning.

Our test set for performance evaluation is the same data set used in [1, 2]. It consisted of 1000 scenarios, each with 100 experts (average connection density 16.05 ± 4.99), 10 topics and a referral network. Our measure of performance is the overall task accuracy of our multi-expert system. For the sake of comparability, for a given simulation across all algorithms, we chose the same sequence of initial expert and topic pairs. For our per-instance query budget, Q, we chose the values 2, 3, and 4, corresponding to single-hop, two-hop and three-hop referrals, respectively. Following [1], our upper bound for single-hop referral is the performance of a network where every expert has access to an oracle that knows the true topic-mean (i.e., $mean(Expertise(e_i, q) : q \in topic_p)\ \forall i, p$) of every expert-topic pair. For two-hop referrals, we use an upper bound based on calculating optimal referral choices up to depth 2. Finally, the baseline is an Expertise-Blind algorithm where the initial expert randomly chooses a connected expert for referral.

The 100 SATenstein solvers we used are obtained by configuring SATenstein2.0 on six well-known SAT distributions (distribution and solver details can be found in [8]). We used the test sets of the SAT distributions as our pool of tasks. Our experiments were carried out on a cluster of dual-core 2.4 GHz machines with 3 MB cache and 32 GB RAM running Linux 2.6.

6 Results

6.1 Performance comparison on synthetic data

Single Hop Referral: For single referral, Table 2 presents the mean task accuracy across the entire data set at specific points of the horizon (samples per subnetwork). For a given horizon, the best performance is highlighted in bold.

Table 2. Performance comparison of referral algorithms with query budget Q = 2

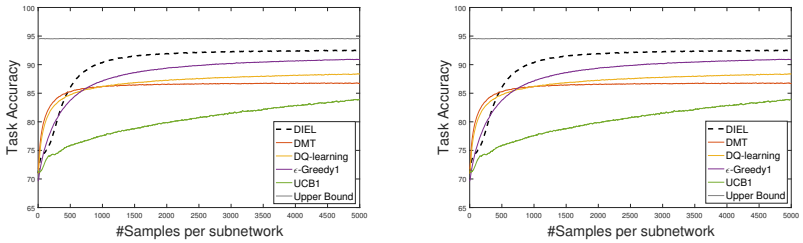
	500	1000	1500	2000	3000	4000	5000
Upper Bound	79.47	79.31	79.27	79.42	79.38	79.41	79.47
DIEL	67.73	**73.35**	**75.35**	**76.33**	**77.33**	**77.76**	**77.96**
DMT	**70.63**	73.06	73.83	74.20	74.54	74.71	74.69
ϵ-Greedy	55.33	56.63	57.80	58.95	60.57	61.95	62.97
ϵ-Greedy1	70.22	72.91	73.97	74.48	74.92	75.19	75.32
UCB1	57.78	59.60	60.80	61.61	63.28	64.49	65.49
UCB2	63.71	64.16	64.19	64.21	64.18	64.19	64.28
UCB-normal	54.38	54.47	54.71	54.97	56.43	58.91	61.39
UCBV	54.99	55.92	56.44	56.87	57.83	58.60	59.15
Q-Learning	65.46	69.19	70.98	72.08	73.46	74.27	74.75
DQ-learning	70.23	72.68	73.74	74.37	75.14	75.60	75.91
Expertise-Blind	54.48	54.46	54.45	54.41	54.48	54.44	54.60

Our results show that except during the very early stages of the simulation, DIEL dominated all the other referral algorithms with a performance approaching the optimal upper bound. A paired t-test reveals that beyond the crossover point (1000 samples per subnetwork), DIEL is better than all other referral algorithms with p-value less than 0.0001. Algorithms with provable performance guarantees may catch up with DIEL given a sufficiently large horizon, but from a practical standpoint, DIEL is an effective referral algorithm to handle real-world scenarios. We extended a random subset of 200 scenarios up to a horizon of 20,000 samples per subnetwork, at which time none of the top performing referral-algorithms from each category had caught up with DIEL.

All referral learning algorithms performed better than our baseline, the expertise-blind referral. Although DQ-learning and ϵ-Greedy1, the best algorithms in the Q-learning and greedy category respectively, obtained a performance close to DIEL, this was conditional to tuning on training data of similar distributional properties. In contrast, DIEL is parameterless and thus does not require additional configuration.

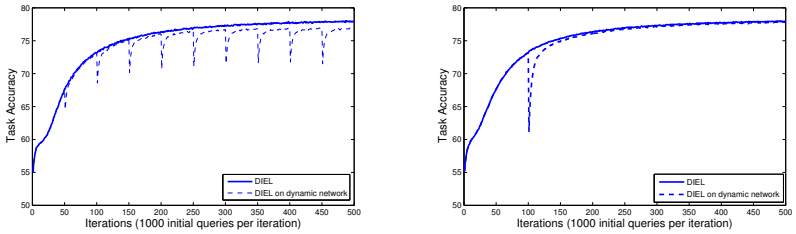
For the remaining results, we retained only the best-performing algorithms in each category, as follows: DIEL (IEL category), ϵ-Greedy1 (Greedy category), UCB1 (UCB category) and DQ-learning (Q-learning category). For comparison, we included additionally, DMT, a horizon-free algorithm.

Multi-hop Referral: In a multi-hop setting, a referred expert can continue referring an instance to another expert as long as the budget permits (excluding cyclic referrals). Figure 1 compares our top-performing referral algorithms with query budget 3 and 4. In Figure 1(a), we compare the performance with an upper bound that calculates optimal choice to depth 2 (optimal choice to depth 1 but the same query budget 3 achieved a task accuracy of 93.05%). Understandably, with a higher query budget, the overall task accuracy of every learning algorithm increases. However, DIEL's rapid performance gain in the early phase of learning



(a) Query budget Q = 3 (b) Query budget Q = 4

Figure 1. Multi-hop referrals



(a) Distributed 5% network change (b) Single point 20% network change

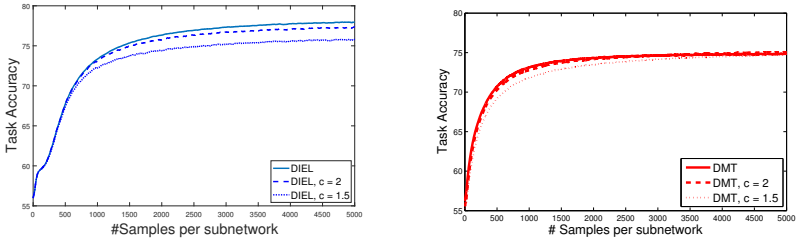
Figure 2. Performance of DIEL with network changes

still enables it to obtain a superior performance. The practical benefit of DIEL against algorithms with theoretical convergence guarantees is particularly evident when compared against UCB1. In fact, DIEL with a lower query budget ($Q = 3$) achieves a better performance than UCB1 with a higher budget ($Q = 4$).

6.2 Robustness of performance

Dynamic network behavior: In practice, referral networks are not static; they evolve over time with new links are forged, experts drop out or new ones join, experts gain or degrade expertise, etc. Here we focused primarily on addition/deletion of experts to the network, both as a one time event with 20% of the experts in the network replaced at iteration 100, and a *distributed change* (modelling more closely a real-world gradual change: 5% of the network changes every 50 iterations). We also ran experiments where the network changes are distributed across time-steps and found qualitatively similar performance.

Figure 2 compares the performance of DIEL on a static network with that on a dynamic network. Our results show that DIEL coped fairly well with a distributed change, and in spite of multiple changes in the network at a regular interval, the final DIEL performance on a dynamic network (task accuracy 76.91%) is slightly worse than DIEL on a static network, but still better than any other referral learning algorithm presented in Table 2. In addition, we ran



(a) Load-balanced DIEL (b) Load-Balanced DMT

Figure 3. Performance of DMT and DIEL for different values of the load-factor c

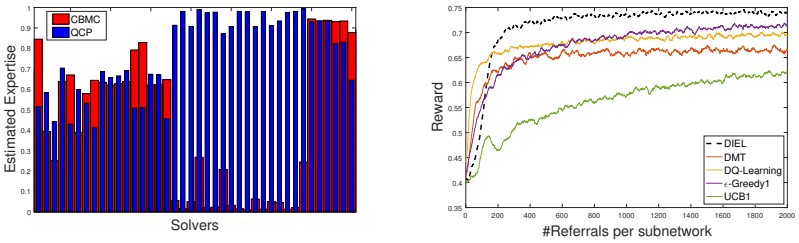
experiments where no experts leave or join, but new referral links get created. Then too, the performance of DIEL proved robust, exhibiting qualitatively similar characteristics. We also found that DIEL could easily cope with a large one time network change (see, Figure 2(b)).

Capacity constraints: Capacity constraints on experts are rarely taken into account in Active Learning (though *Proactive Learning* [14] considers similar aspects). In reality, of course, experts can handle only a limited number of tasks at any given time, and the capacity of the best experts can easily be exceeded. This was borne out by our simulations – for our DIEL simulations, expertise and load were correlated with correlation coefficient $r = 0.69$.

We simulated transient (bursty) overloading. Let $load(e_i, m)$ denote the number of tasks expert e_i received among the last m tasks (*initial* or *referred*) the network received. In a network of k experts, a fair load for every expert is $\frac{m}{k}$. An expert is overloaded if $load(e_i, m) \geq c * \frac{m}{k}$, where the *l*oad-factor $c > 1$. In our experiments we assumed that an expert reaching her load limit becomes unavailable until completing one or more current tasks. Even with a tight value of $c = 1.5$, we find that the performance of the referral-algorithms degrades gracefully, and surprisingly, sometimes causing a performance improvement because of forced exploration as tasks are sent to other experts. For example, as shown in Figure 3(b), the load-balanced version of DMT with a load-factor of 2 slightly outperforms DMT without any capacity constraint. That we observed a graceful performance degradation with all the referral algorithms leads us to conjecture that load balancing is facilitated by the distributed nature of the learning setting.

6.3 SATenstein SLS solvers as experts

So far, we have presented our results on synthetic data and binary rewards. Here, we describe our results where experts are Stochastic Local Search (SLS) solvers and the task is to solve a SAT problem instance. In addition to the attractive properties of SAT solvers listed in [9] (e.g, easy availability of a large number of experts with differential expertise (Figure 4(a)), and the straight-forward verifiability of solutions), they allow us to easily express solution quality as a



(a) Skill estimates (b) Performance comparison

Figure 4. Expertise estimates of a subset of solvers on background data of two SAT distributions and performance comparison with `SATenstein` solvers as experts

function of run time, allowing us to test the referral algorithms under continuous rewards.

In these experiments, in order to save computational cycles, we solely focus on the referral learning behavior of the network; i.e., we assume that the initial expert always refers a task to a connected expert. We set the budget C for solving each instance to 1 CPU second, which is the maximum time in which, on a similar computing architecture, configured high-performance `SATenstein` solvers were found to solve a majority of the instances in their expertise area [8] (This was corroborated in our experiments). The reward is computed as (C - r_t) where r_t is the run time (when a solver fails to solve an instance, $r_t = 1$). With C set to 1 in our experiments, the reward is bounded by [0, 1) with a failed task fetching a reward of 0 and higher rewards implying faster solutions. So in this setting, through continuous reward, we have incorporated solution quality (in this case, run time) in our experiments.

Figure 4(b) presents the performance comparison of referral-learning algorithms where experts are SAT solvers and topics are SAT problem distributions on 10 randomly chosen referral networks. We found that `DIEL` outperformed all other algorithms, with `DMT`, `DQ-learning`, and ϵ-`Greedy1` achieving a performance close to `DIEL` (even when we extended the runs to 4000 referrals per subnetwork for ϵ-`Greedy1`, it had not yet caught up with `DIEL`). Similar to the results obtained on our synthetic data, we found that `UCB` had the slowest rate of improvement in the initial stage of learning. These results highlight the following. First, even with real experts, a well-defined task and very few distributional assumptions on expertise, it is possible to learn effective referral choices. Second, `DIEL`'s superiority over other referral-learning algorithms is not just restricted to synthetic data, nor dependent on binary rewards.

Acknowledgements: This research is partially funded by the National Science Foundation grant EAGER-1649225.

Bibliography

[1] KhudaBukhsh, A.R., Jansen, P.J., Carbonell, J.G.: Distributed Learning in Expert Referral Networks. In: European Conference on Artificial Intelligence (ECAI), 2016. (2016) 1620–1621

[2] KhudaBukhsh, A.R., Carbonell, J.G., Jansen, P.J.: Proactive Skill Posting in Referral Networks. In: Australasian Joint Conference on Artificial Intelligence, Springer (2016) 585–596

[3] Agrawal, R.: Sample mean based index policies with o (log n) regret for the multi-armed bandit problem. Advances in Applied Probability (1995) 1054–1078

[4] Lai, T.L., Robbins, H.: Asymptotically efficient adaptive allocation rules. Advances in applied mathematics **6**(1) (1985) 4–22

[5] Audibert, J.Y., Munos, R., Szepesvári, C.: Tuning bandit algorithms in stochastic environments. In: International Conference on Algorithmic Learning Theory, Springer (2007) 150–165

[6] Abdallah, S., Lesser, V.: Learning the task allocation game. In: Proc of AAMAS '06, ACM (2006) 850–857

[7] Zhang, C., Lesser, V., Shenoy, P.: A Multi-Agent Learning Approach to Online Distributed Resource Allocation. In: Proc. of IJCAI-09. Volume 1., Pasadena, CA (2009) 361–366

[8] KhudaBukhsh, A.R., Xu, L., Hoos, H.H., Leyton-Brown, K.: SATenstein: Automatically building local search SAT solvers from components. Artificial Intelligence **232** (2016) 20–42

[9] KhudaBukhsh, A.R., Carbonell, J.G., Jansen, P.J.: Proactive-DIEL in Evolving Referral Networks. In: European Conference on Multi-Agent Systems, Springer (2016)

[10] Donmez, P., Carbonell, J.G., Bennett, P.N.: Dual Strategy Active Learning. Machine Learning ECML 2007 (2007) 116–127

[11] Watkins, C.J., Dayan, P.: Q-Learning. Machine Learning **8**(3-4) (1992) 279–292

[12] Hasselt, H.V.: Double Q-Learning. In: Advances in Neural Information Processing Systems. (2010) 2613–2621

[13] Auer, P., Cesa-Bianchi, N., Fischer, P.: Finite-time analysis of the multiarmed bandit problem. Machine Learning **47**(2-3) (2002) 235–256

[14] Donmez, P., Carbonell, J.G.: Proactive Learning : Cost-Sensitive Active Learning with Multiple Imperfect Oracles. Proceedings of CIKM '08 **08** (2008) 619–628