Reinforcement Learning in Distributed Domains: Beyond Team Games

David H. Wolpert
NASA Ames Research Center
Moffett Field, CA 94035
dhw@ptolemy.arc.nasa.gov

Joseph Sill
Ripfire, Inc.
San Fransisco, CA 94102
joe@ripfire.com

Kagan Tumer
NASA Ames Research Center
Moffett Field, CA 94035
kagan@ptolemy.arc.nasa.gov

Abstract

Using a distributed algorithm rather than a centralized one can be extremely beneficial in large search problems. In addition, the incorporation of machine learning techniques like Reinforcement Learning (RL) into search algorithms has often been found to improve their performance. In this article we investigate a search algorithm that combines these properties by employing RL in a distributed manner, essentially using the team game approach. We then present bi-utility search, which interleaves our distributed algorithm with (centralized) simulated annealing, by using the distributed algorithm to guide the exploration step of the simulated annealing. We investigate using these algorithms in the domain of minimizing the loss of importance-weighted communication data traversing a constellations of communication satellites. To do this we introduce the idea of running these algorithms "on top" of an underlying, learning-free routing algorithm. They do this by having the actions of the distributed learners be the introduction of virtual "ghost" traffic into the decision-making of the underlying routing algorithm, traffic that "misleads" the routing algorithm in a way that actually improves performance. We find that using our original distributed RL algorithm to set ghost traffic improves performance, and that bi-utility search — a semi-distributed search algorithm that is widely applicable — substantially outperforms both that distributed RL algorithm and (centralized) simulated annealing in our problem domain.

1 Introduction

Use of a centralized algorithm to perform search in large spaces can be quite problematic. This is especially so when the search problem is to find a control policy for a large, distributed Multi-Agent System (MAS) in which communication limitations restrict the amount of data that a centralized algorithm could exploit. Satisfactory performance in such domains may not be possible at all unless one uses a distributed rather than centralized algorithm.

A canonical example of such a control problem, central to mid-term NASA interplanetary missions, is controlling the communication of scientific data up from a planet, across an orbiting constellation of communication satellites orbiting that planet, and thence down to Earth. In addition to its distributed nature, this domain presents a number of other difficulties. One is that the satellites must be able to exercise a large degree of autonomy, since communication with earth can take hours. Another is the need for robustness and adaptability, since the communication loads can be quite unpredictable. Both of these difficulties argue for the use of learning algorithms to perform the distributed control. In particular, the technique of Reinforcement Learning (RL) has often been successfully applied to search algorithms [6; 14; 26], and suggests itself for this domain. More generally, integrating such RL into a distributed algorithm may be highly beneficial for many distributed control problems.

In this article we concentrate on algorithms that achieve this integration by having the agents in the MAS each run RL algorithms. The design problem is how to ensure that as those agents each try to optimize the values of their personal utility functions, an overall "world utility function", quantifying the desirability of the various possible behaviors of the overall system, is maximized. Though most naturally viewed as a means of distributed control, such an approach can actually be used for distributed search in general, by identifying each of the distributed components of the search problem with an agent.

The COllective INtelligence (COIN) framework addresses such design problems [20; 23; 24]. That work has focussed on modifying the agents' personal utility functions so that they induce the best possible value of the world utility function. The learning-based search algorithms we have explored to date based on the COIN framework have been extremely successful. In fact, even in domains where centralized algorithms can be used, due to their adaptability and their exploiting the learn-

ing power of each of the individual agents COIN-based systems typically perform better than such centralized algorithms.

Nonetheless, many centralized algorithms have a number of well-understood strengths, that if integrated with a distributed RL approach like that of COINs might result in performance better than either approach alone. In addition, often search algorithms that are conventionally viewed as centralized can be implemented in a quasi-distributed manner. This opens up the possibility that a hybrid of such an algorithm with a distributed RL algorithm might not only achieve better performance than either alone, but also be implementable in a large distributed problem domain.

In this paper we investigate hybrids of distributed RL algorithms and search algorithms that are (conventionally viewed as) centralized. We concentrate on the use of Simulated Annealing (SA) as the centralized algorithm. SA is a natural choice because its behavior and strengths are very well-understood [1; 4; 5; 13; 16; 17; 21] In addition, at the expense of periodic centralized broadcasts to all agents of whether to choose a new action or repeat their most recent one, and of associated measurements of global performance, SA can be implemented in a quasi-distributed manner. No inter-agent communication is required beyond those broadcasts and measurements.

As our problem domain in this paper we concentrate on the problem of minimizing the importance-weighted loss of scientific data flowing across a constellation of communication satellites. The first contribution of this paper is a novel "baseline" distributed control algorithm for this domain, one that involves no learning. To minimize the number of confounding distinctions between that baseline algorithm and the COIN algorithm we investigate, we have that COIN algorithm "run on top" of the baseline algorithm. More precisely, our second contribution is to use a baseline algorithm in concert with a COIN algorithm in which each agent's action is the determination of fictitious "ghost traffic" that is presented to the baseline algorithm, thereby (hopefully) inducing that baseline algorithm to achieve an even better value of the world utility. (Note that this idea can be applied with most any baseline algorithm and most any distributed RL algorithm.) In this paper we investigate the simplest kind of COIN algorithm, which essentially works by using the team game (TG) approach [8]. Our final contribution is the hybrid of that distributed RL algorithm with SA, a hybrid we call "bi-utility search". This hybrid works by interleaving the Boltzmann-distribution-based exploitation step of conventional SA with a novel exploration step, where rather than have the overall system randomly step away from the current point each agent takes a step that its RL algorithm recommends.

In the next section we discuss the use of distributed RL in the context of our particular problem domain. In Section 3 we define that problem domain in a fully formal manner. Then in Section 4 we present experimental comparisons of TG-based bi-utility search with both TG and SA in that domain. We find that using our original distributed RL algorithm to set ghost traffic improves performance, and that bi-utility search — a semi-distributed search algorithm that is widely applicable — substantially outperforms both that distributed RL algorithm and (centralized) simulated annealing in our problem domain.

2 Background

Many future NASA projects will involve data traversing a constellation of communication satellites. In such a constellation, each satellite continually receives data (e.g., data uplinked from a planet, relayed from another satellite, or even directly observed), and needs to relay this data along a path back to Earth. In general, different data are likely to have different levels of importance. Accordingly, a suitable overall goal for such a constellation to minimize is the total importance-weighted loss of data across the network. We can cast this as maximizing a world utility function given by the negative of that loss. Due to various hardware limitations (e.g., storage, power, bandwidth), even at those times that a particular satellite has a direct link to Earth, to maximize this world utility, it will often still be preferable for that satellite to route its current data across other satellites rather than send it directly to Earth For example, since we are implicitly assuming multiple communication transmitters at each satellite, it may benefit a satellite to drain its current disk by offloading data to other satellites while it is also downloading data to Earth. In this way, the disk will be emptier and therefore better able to accommodate future data surges.

Clearly given the time delays and hardware limitations inherent in this problem, one would like to use as decentralized an algorithm as possible for determining the routing. Unfortunately, while they can be implemented in decentralized forms, traditional routing algorithms (e.g., shortest path algorithms [2; 3; 10]) are ill-suited for our world utility function. Accordingly, in this study we had to first develop a non-learning "baseline" distributed routing algorithm, one that we expect will do well for our world utility. To create a distributed RL version of that algorithm we then consider the use of "ghost" traffic, which is non-existent traffic that is presented along with the real traffic to the baseline routing algorithms running on the satellites. The goal of the learning is to find such ghost traffic that will "fool" the baseline algorithm into having the satellites take better routing decision (as far as world utility is concerned) than they would otherwise.

A natural choice for the type of learning to use to optimize ghost traffic is RL [6; 8; 12; 15; 19; 26]. Indeed, distributed MAS control algorithms in which the agents independently attempt to maximize their personal utilities using RL have been successfully used to optimize world utility in several large decentralized problems [11; 14; 18; 20; 22; 23; 24]. In the TG approaches in par-

ticular [8], every agent uses RL with its utility set to the world utility, so the only centralized communication needed is periodic broadcasts of the world reward signal (in our case presumably calculated on Earth where all the data is collected), a signal shared by all agents. This contrasts with conventional centralized RL, in which the information periodically broadcast has to be "personalized" to each satellite, being that satellite's updated routing policy.

There are a number of major shortcomings of conventional TG however. One of them is that for large systems each agent faces an overwhelming signal-to-noise problem in determining how its actions affect the rewards it receives [25]. Another arises from the fact that each agent's predictions of expected rewards is independent of the other agents' actions, i.e., each agent i considers a distribution of the form:

$$P(utility \mid action_i; observation_i)$$

rather than one of the form

$$P(utility, \mid action_1, action_2, ...; observation_i).$$

Unfortunately, it is quite common that utility is maximized at one joint action profile $(action'_1, action'_2, ...)$, but that due to the historical probability distribution over action profiles, $action'_i$ does not maximize $E(utility, \mid action_i; observation_i)$, and therefore is not taken by agent i. In essence, there is a synchronization problem.

To simplify matters, we consider TG using the simplest possible RL algorithm, in which each learner makes no observations and uses no lookahead, choosing its action (ghost traffic level) at any moment by sampling a Gibbs distribution[1] over its estimates for the associated rewards [7; 23]. For stationary traffic patterns, with such RL algorithms the natural performance measure is given by the function taking joint-action profiles to the world reward.

One distributed approach to such a search problem that avoids the synchronization difficulty of TG while still only requiring the centralized broadcast of information that isn't personalized is simulated annealing (SA) [1; 9]. The variant of SA considered here repeats a two-step process. In the decentralized "exploration step", all agents make (usually small) random changes in their action. In the subsequent centralized "exploitation step", a Gibbs distribution over the associated rewards is sampled to choose between the new action profile and the previous one. That (non-personalized) choice — use your new action or your previous one — is then broadcast out to all the agents. (Note that such a broadcast is no more personalized than the world reward value that must be similarly broadcast in TG.) The major disadvantage of SA compared to distributed RL is that only one "learner" is exploited in SA, at the central location.

[1] I.e., choose action a_i with estimated reward R_i with probability $p(a_i) = \frac{\exp(-R_i/T)}{\sum_j \exp(-R_j/T)}$, where T determines the "temperature" parameter which controls the exploration/exploitation tradeoff.

In this article we introduce "bi-utility search", which is a hybrid of SA and distributed RL that combines the strengths of both. Bi-utility search is identical to SA, except that the random perturbation of each agent's action occurring in the exploration step is replaced by a perturbation determined by the RL algorithm of that agent. Since the exploitation of SA is used, the synchronization problem of distributed RL is avoided. On the other hand, since the exploration of distributed RL is used, the blind random nature of SA is replaced by a (hopefully) more intelligent RL-guided process. Particularly in large domains, we would expect that that guiding might provide major improvements in performance.

3 Constellations of Satellites

In our simulations each **satellite** i has a storage **capacity** c_i, measured in amount of data. It also has a **bandwidth** (also measured in amount of data, implicitly per unit time interval) b_{ik} to each other satellite k. Earth is simply a special satellite with $c_0 = \infty$. At each time step t, an amount of **new data** y_{ijt} of importance j is introduced to the system at satellite i. We add y_{ijt} to the total amount of data of importance j received by i from all other satellites at t and then to the amount of unsent data on the disk left over from the previous time step to get the total **volume** of data of importance j at i at t, v_{ijt}. If $\sum_j v_{ijt} > c_i$, then that difference constitutes the amount of data lost at satellite i at t. We assume that the same proportion of data is dropped for each importance level, since once the disk is full the satellite is unable to examine any data sent to it and determine its priority.

Define l_{ijt} to be the amount of data of importance j **lost** (i.e., dropped) at satellite i at t. Define the associated cost as $l_{ijt} w_j$ for some data-importance**weights** w_j. Then the objective function we wish to maximize is the time-average of

$$G_t \equiv 1 - \frac{\sum_{ij} w_j l_{ijt}}{\sum_{ij} w_j y_{ijt}} \; . \tag{1}$$

Denote a path by a sequence of satellites $\vec{s} \equiv s_1, \ldots, s_p$, where s_1 represents the originating satellite and s_p is the satellite which ultimately sends the packet to Earth. At each t each satellite i evaluates a potential decision to send some data to satellite k by estimating the "headroom" of the optimal path to Earth beginning at k. The headroom of path $\vec{s}$ is the available room for additional data, given the available storage capacity on each satellite along $\vec{s}$ and the bandwidth of each connection along $\vec{s}$. Formally, the headroom $H(\vec{s})$ of a path $\vec{s}$ is given by:

$$H(\vec{s}) = \min_{q \in \{1, \ldots p\}} (\min(b_{s_q, s_{q+1}}, c_{s_{q+1}} - \sum_j v_{s_{q+1}, j, t})) \; . \tag{2}$$

The presumption is that a path that currently has headroom should be favored over one with low headroom, since the likelihood of data being dropped is lower. This

is just like how a path with low current cost is favored over a path with high current cost in traditional Shortest Path (SP) data routing [2; 3]).

Note that in a real system, a particular satellite i does not have access to the precise $v_{jkt}, j \neq i$ at the current t. Hence the current headroom values would have to be estimated by satellites (just as current cost values are estimated in SP routing). Because we are interested in how to improve the performance of the base algorithm, in these experiments we ignore this issue and supply each satellite with the current v_{jkt}. (The estimation of the v_{jkt} would introduce a systematic error that should not affect the ranking of the algorithms discussed below).

It is straightforward to calculate the maximum headroom path from each satellite to Earth using a version of Dijkstra's shortest path algorithm [3; 10]. Let H_{ijt} be the time t headroom of the optimal path originating at satellite i and with the first hop being to satellite j. The satellite at which data originates does not directly decide on the full path to Earth taken by the data it transmits; it simply decides on the first hop in the path and sends its data to the appropriate satellite based on the H_{ijt}'s. (Similarly, in traditional data routing, a router only selects the first hop along the seemingly shortest path, based on the costs). More precisely, define

$$j^*_{it} \equiv \text{argmax}_j H_{ijt}. \quad (3)$$

If $H_{ij^*_{it}} > v_{ikt}$, then all of v_{ikt} $\forall k$ is sent to satellite j^*_{it} and $H_{ij^*_{it}}$ is updated by subtracting v_{ikt} off of the headroom estimate to reflect the fact that that much data has already been sent to that satellite. If $H_{ij^*_{it}} < v_{ikt}$, then an amount $H_{ij^*_{it}}$ is sent (highest importance data first) to j^*_{it} and $H_{ij^*_{it}}$ is updated to equal zero.

This procedure is then repeated until either:

1. $v_{ikt} = 0$ $\forall k$, or
2. $H_{ijt} = 0$ $\forall j$.

If the second condition occurs before all data has been routed, then the remaining data is not sent anywhere and instead kept on the disk until the next iteration in the hopes of routing it successfully then.

While this routing algorithm performs respectably, it is susceptible to the same phenomena that hamper traditional SP routing [20]: the satellites do not explicitly act to optimize G, and can therefore potentially work at cross-purposes. To alleviate this we introduce additive perturbations δ_{ijt} to the headroom estimates H_{ijt} and then perform the routing according to the perturbed headroom estimates. The δ_{ijt} are free parameters to be determined via an optimization algorithm, and because their effects on the headroom is the same as that of actual data, we call them "ghost" traffic. Our goal then is to find the $\{\delta_{ijt}\}$ at each t that optimize the time-average of $G_t(\{\delta_{ijt}\})$ — a search problem. In this paper we consider performing that search via two variants of simulated annealing, team game reinforcement learning and bi-utility search.

4 Experimental Results

In our experiments we had three importance levels ($w_j \in \{1, 2, 3\}$). For each satellite, for each importance level, at each t, new traffic was introduced by uniformly sampling [0.0, 0.5]. We used a network of moderate connectivity with 20 satellites altogether (150 δ's for each t). For the purposes of learning, we associated the actions of an "agent" at t with the setting of δ_{ijt} for a particular ij pair.

As mentioned before, to focus on the goals of the individual RL-based agents' rather than on how they try to achieve those gaols, we used the simplest possible RL algorithms in the TG agents. Since they make no observations, formally each such an agent has one "state". Each agent's possible actions (ghost traffic on a particular link) are chosen from the set of discrete values {-5, -4, ..., 4}. To reflect the nonstationarity in the environment, each agent applies proportional updating to average its empirically collected training data, giving an estimated reward value for each candidate action. (We had the geometric updating factor set to .1 always.) The RL algorithm then decides what action to take by sampling a Gibbs distribution over those associated reward estimates (cf. Section 1), using a decaying temperature parameter. Temperatures annealed linearly from .1 down to .01 after 5000 τ intervals had passed.

We investigate five overall search algorithms. Each of them uses "time steps", labeled by τ, that consist of 200 consecutive t values of the underlying system. "G_τ" for each such "time step" is then defined to be the average G_t over that interval. (Note that successive values of G_τ are not statistically independent; some "leakage" of system behavior will occur across the boundary between successive τ.) We took δ to be constant across each such time step.

- **Baseline:** Algorithm outlined in Section 3; no learning.
- **Team Games (TG):** Each agent uses the RL algorithm described above to try to independently maximize G_τ at each τ.
- **Simulated Annealing (SA):** A two-step process is iterated. First the δ matrix of the previous time step is perturbed (uniformly) randomly and the resulting δ matrix is implemented in the system. The ensuing G_τ is measured (exploration), and at the next τ that vector is probabilistically "accepted or rejected" in favor of the pre-perturbation vector, by sampling a Gibbs distribution over the two associated recorded G_τ values (exploitation). The vector so chosen is implemented, the ensuing G_τ measured, and the process repeats.
- **"Localized" Simulated Annealing:** Simulated annealing modified by restricting each component of the new perturbation vector to be at most one bin away from the old one.
- **G-Based Simulated Annealing (bi-utility search):** Localized simulated annealing modified

by having each component of the new perturbation vector set by having the associated TG agent choose among the 3 candidate bins.

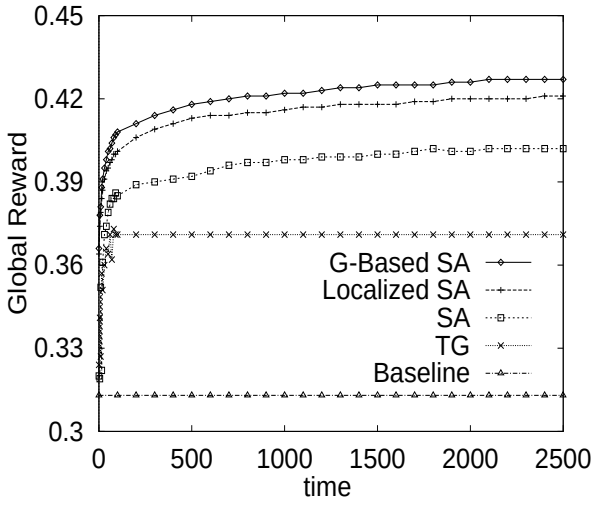

Figure 1: Overall performance of the algorithms in the communication satellites problem

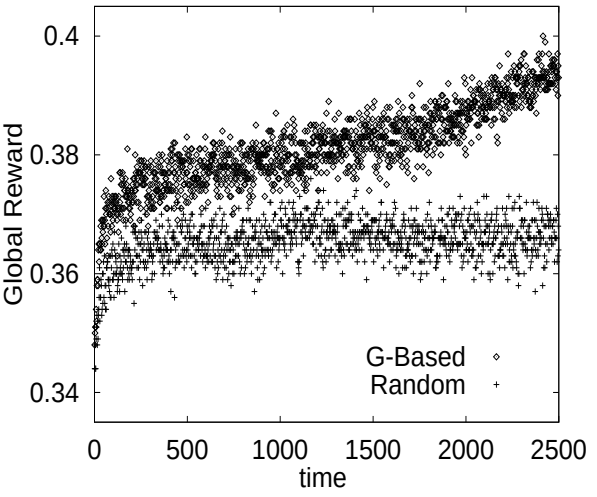

Figure 2: Exploration step performance for localized and G-guided simulated annealing.

Figure 1 compares the performance of these five algorithms on the network described above, averaged over 100 runs (the resulting error bars are too small to depict). The team game RL algorithm performs better than the baseline algorithm. However, the simulated annealing results show that TG is falling prey to problems like that of synchronization. When SA's minimal need for centralized broadcast information can be met, it should be used rather than TG. Next, note that the localized version of of SA significantly outperforms standard SA, reflecting the inherent smoothness of the surface being searched. Finally, while both localized SA and bi-utility search are superior to both TG and SA, the G-guided version (bi-utility search) is the clear winner. This demonstrates that that algorithm does successfully combine the strengths of simulated annealing and reinforcement learning.

Figure 2 shows the G_τ of the δ's generated by the exploration steps of the two algorithms. RL-based exploration clearly produces better states, and this gap in performance increases in time. This translates into its higher performance.

5 Discussion

In search problems over large spaces, even if it is physically possible to use a centralized search algorithm, often one might expect that better performance would be achieved using a distributed rather than a centralized algorithm, due to its parallelization advantage. This raises the issue of what algorithms to run on each of the "agents" in such a distributed algorithm. Recent work has demonstrated that Reinforcement Learning (RL) can be quite gainfully applied to centralized search algorithms [6; 14; 26]. Here we investigate extending RL into a distributed setting.

We start by presenting a distributed RL search algorithm very similar to Team Games (TG) [8; 23]. In particular, we present experiments showing how to use this algorithm to determine the "ghost traffic" to be introduced into a pre-fixed algorithm designed for routing communication data across a constellation of satellites, in the hopes of fooling that algorithm into performing better. Our experiments indicate that this distributed RL algorithm significantly reduces the importance-weighted amount of data lost in such a domain.

Unfortunately, TG has many shortcomings, including a "signal-to-noise" problem and a "synchronization" problem. The first problem can be addressed using the COllective INtelligence (COIN) framework [20; 23; 24]. The second can be obviated if (non-personalized) periodic broadcast signals are allowed. In particular, Simulated Annealing (SA) only uses such signals, and avoids the synchronization problem. In fact, we find that SA outperforms TG in our problem domain.

We introduce bi-utility search as a way to exploit the parallelization advantage of a distributed algorithm without incurring the difficulties of TG. This procedure interleaves a distributed algorithm (e.g., TG) with SA, by using the distributed algorithm to govern an exploration step, which is followed by the exploitation step of SA. Our experiments show that TG-based bi-utility search substantially outperforms both TG and SA in our constellation of communication satellites domain.

As investigated here, bi-utility search uses Gibbs sampling both for the exploration and exploitation steps. It also uses world reward in both of those Gibbs samplers. None of that is *a priori* necessary; bi-utility search is a general approach for combining a centralized exploitation step with a decentralized exploration step. In par-

ticular, the distributed algorithm can be based on the COIN framework, and thereby avoid the signal-to-noise problem of TG. Preliminary results not reported here indicate that such COIN-based bi-utility search performs even better than TG-based bi-utility search. We are currently investigating these issues further.

References

[1] E. Aarts and J. Korst. *Simulated Annealing and Boltzmann Machines: A Stochastic Approach to Combinatorial Optimization and Neural Computing*. John Wiley and Sons, 1989.

[2] R. E. Bellman. On a routing problem. *Quarterly of Applied Mathematics*, 16:87–90, 1958.

[3] D. Bertsekas and R. Gallager. *Data Networks*. Prentice Hall, Englewood Cliffs, NJ, 1992.

[4] K. D. Boese. *Models for Iterative Global Optimization*. PhD thesis, University of California, Los Angeles, 1996.

[5] K. D. Boese, A. B. Kahng, B. A. McCoy, and G. Robins. A new adaptive multi-start technique for combinatorial global optimizations. *Operations Research Letters*, 16(2):101–113, 1994.

[6] J. A. Boyan and A. Moore. Learning evaluation functions for global optimization and boolean satisfiability. In *Proceedings of the Fifteenth National Conference on Artificial Intelligence*. AAAI Press, 1998.

[7] C. Claus and C. Boutilier. The dynamics of reinforcement learning cooperative multiagent systems. In *Proceedings of the Fifteenth National Conference on Artificial Intelligence*, pages 746–752, Madison, WI, June 1998.

[8] R. H. Crites and A. G. Barto. Improving elevator performance using reinforcement learning. In D. S. Touretzky, M. C. Mozer, and M. E. Hasselmo, editors, *Advances in Neural Information Processing Systems - 8*, pages 1017–1023. MIT Press, 1996.

[9] R. Diekmann, R. Luling, and J. Simon. Problem independent distributed simulated annealing and its applications. In *Applied Simulated Annealing*, pages 17–44. Springer, 1993.

[10] E. Dijkstra. A note on two problems in connection with graphs. *Numeriche Mathematics*, 1(269-171), 1959.

[11] J. Hu and M. P. Wellman. Multiagent reinforcement learning: Theoretical framework and an algorithm. In *Proceedings of the Fifteenth International Conference on Machine Learning*, pages 242–250, June 1998.

[12] L. P. Kaelbing, M. L. Littman, and A. W. Moore. Reinforcement learning: A survey. *Journal of Artificial Intelligence Research*, 4:237–285, 1996.

[13] S. Kirkpatrick, C. D. Jr Gelatt, and M. P. Vecchi. Optimization by simulated annealing. *Science*, 220:671–680, May 1983.

[14] M. L. Littman and J. Boyan. A distributed reinforcement learning scheme for network routing. In *Proceedings of the 1993 International Workshop on Applications of Neural Networks to Telecommunications*, pages 45–51, 1993.

[15] R. Moll, A. G. Barto, and T. J. Perkins. Learning instance-independent value functions to enhance local search. In *Advances in Neural Information Processing Systems - 11*, pages 1017–1023. MIT Press, 1999.

[16] D.T. Pham and D. Karaboga. *Intelligent Optimization Techniques*. Springer, London, UK, 2000.

[17] C. P. Robert and G. Casella. *Monte Carlo Statistical Methods*. Springer-Verlag, New York, 1999.

[18] T. Sandholm and R. Crites. Multiagent reinforcement learning in the iterated prisoner's dilemma. *Biosystems*, 37:147–166, 1995.

[19] R. S. Sutton and A. G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, Cambridge, MA, 1998.

[20] K. Tumer and D. H. Wolpert. Collective intelligence and Braess' paradox. In *Proceedings of the Seventeenth National Conference on Artificial Intelligence*, pages 104–109, Austin, TX, 2000.

[21] R. V. V. Vidal, editor. *Applied Simulated Annealing (Lecture Notes in Economics and Mathematical Systems)*. Springer, 1993.

[22] D. H. Wolpert, S. Kirshner, C. J. Merz, and K. Tumer. Adaptivity in agent-based routing for data networks. In *Proceedings of the fourth International Conference of Autonomous Agents*, pages 396–403, 2000.

[23] D. H. Wolpert and K. Tumer. An Introduction to Collective Intelligence. Technical Report NASA-ARC-IC-99-63, NASA Ames Research Center, 1999. URL:`http://ic.arc.nasa.gov/ic/projects/coin_pubs.html`. To appear in Handbook of Agent Technology, Ed. J. M. Bradshaw, AAAI/MIT Press.

[24] D. H. Wolpert, K. Tumer, and J. Frank. Using collective intelligence to route internet traffic. In *Advances in Neural Information Processing Systems - 11*, pages 952–958. MIT Press, 1999.

[25] D. H. Wolpert, K. Wheeler, and K. Tumer. Collective intelligence for control of distributed dynamical systems. *Europhysics Letters*, 49(6), March 2000.

[26] W. Zhang and T. G. Dietterich. Solving combinatorial optimization tasks by reinforcement learning: A general methodology applied to resource-constrained scheduling. *Journal of Artificial Intelligence Reseach*, 2000.