Chapter 8
Conclusions

Reinforcement learning offers the tantalizing promise of software systems that autonomously improve their performance on sequential decision-making tasks, according to the following methodology:

1. Collect data by observing simulations of the task.
2. Statistically learn to predict the long-term outcomes of the chosen decisions.
3. Use those predictions as an evaluation function to guide future decisions with great foresight.

In this dissertation, I have described and analyzed a practical method for applying this methodology to general combinatorial optimization tasks. I have also introduced several new algorithms for efficient reinforcement learning in large state spaces.

This concluding chapter first reviews the dissertation's scientific contributions to the state of the art in reinforcement learning, heuristic search, and global optimization. It then outlines a number of promising directions for future research in learning evaluation functions.

8.1 Contributions

The principal contributions of this thesis may be summarized as follows:

- (Chapter 2) I have introduced **two novel value-function-approximation algorithms**: Grow-Support for deterministic problems (§2.2) and ROUT for stochastic acyclic problems (§2.3). Their primary innovation is that, without falling prey to the curse of dimensionality, they are able to explicitly represent which states are already solved and which are not yet solved. Using this information, they "work backwards," computing accurate V^* values at targeted unsolved states. By treating solved and unsolved states differently, Grow-Support and ROUT eliminate the possibility of divergence caused by repeated value re-estimation.

- (§3.1.3) In the context of global optimization, I have recognized that **additional features** of each state, other than the state's objective-function value, can provide useful information for decision making in search. Traditional algorithms either ignore this information or incorporate it in an *ad hoc* manner. STAGE provides a principled, automatic mechanism for exploiting additional state features.

- (§3.2) I have defined the predictive **value function of a local search procedure**, $V^{\pi}(x)$; described the conditions under which it corresponds to the value function of a Markov chain; and described how it may be learned from simulation data by a function approximator.

- (§3.2) I have introduced **STAGE**, a straightforward algorithm for exploiting the learned approximation of V^{π} to guide future search. STAGE is general: it can be applied to any optimization problem to which hillclimbing applies. It may be viewed as an adaptive multi-restart approach to optimization; the adaptive component is automatically learned from simulation data on each problem instance.

- (§3.3) I have presented **two illustrative domains**—the 1-D wave minimization example and the bin-packing example—and demonstrated how STAGE succeeds on them. These examples provide clear intuitions of how learning evaluation functions can improve search performance.

- (§3.4) I have analyzed the **theoretical conditions** under which STAGE is well-defined and efficient. Specifically, I have shown that V^{π} is well-defined for *any* local search procedure π, as long as the objective function is bounded below; however, STAGE learns to approximate V^{π} most efficiently if π is proper, Markovian, and monotonic. I have provided several methods for converting improper and nonmonotonic procedures into a form suitable for STAGE.

- (Chapters 4 and 5) I have contributed **empirical evidence** that STAGE is applicable, practical and effective on a wide variety of large-scale optimization tasks. On most tested instances, STAGE outperforms both multi-start hillclimbing and a good implementation of simulated annealing. On challenging instances of the Boolean satisfiability task, STAGE successfully learned from WALKSAT, a non-greedy local search procedure—and produced the best published solutions to date. The empirical analyses of Chapter 5 demonstrate STAGE's overall robustness with various parameter settings, and also explain under what circumstances STAGE will fail to improve optimization performance.

- (§6.1) I have introduced a **least-squares formulation of the TD($\boldsymbol{\lambda}$) algorithm**, extending the work of Bradtke and Barto [96]. I empirically demonstrate the improved data efficiency of this formulation, and give a new intuitive explanation for the source of this efficiency: the statistics kept by LSTD(λ) amount to a compressed *model* of the underlying Markov process.

- (§6.2) I have motivated, described and tested a new approach to **transferring learned information** from previously solved optimization problem instances to new ones. By using a voting mechanism, the X-STAGE algorithm avoids having to normalize the objective function across disparate instances.

- (Chapter 7) I have **surveyed the literatures** of several related areas: value function approximation (§2.1.2, §7.2); adaptive multi-restart techniques for local search (§7.1), including genetic algorithms (§7.4); and simulation-based learning methods for improving AI search (§7.3). Taken together, these surveys provide a useful collection of references for researchers interested in automatic learning and tuning of evaluation functions.

8.2 Future Directions

The main conclusion of this thesis is that learning evaluation functions can improve global optimization performance. STAGE is a simple, practical technique that demonstrates this. STAGE's simplicity enables many potentially useful extensions; I suggest some of these in Section 8.2.1. Beyond STAGE, there are at least two conceptually different ways of utilizing value function approximation in global optimization; I discuss these in Section 8.2.2. Finally, in Section 8.2.3, I consider the potential for learning evaluation functions by non-VFA-based, *direct meta-optimization* methods.

8.2.1 Extending STAGE

Many modifications to and extensions of STAGE, such as varying the regression model and training technique used to approximate V^π, have been investigated in Chapters 5 and 6 of this thesis. However, many further interesting modifications remain untried. I describe several of these here:

Non-polynomial function approximators. My study of Section 5.2.2 was limited to first- through fifth-order polynomial models of V^π. It would be interesting to see whether other linear architectures—such as CMACs, radial basis function networks, and random-representation neural networks—could produce better fits and better performance.

A more ambitious study could investigate efficient ways to use *nonlinear architectures*, such as multi-layer perceptrons or memory-based fitters, with STAGE. In the context of transfer, the training speed of the function approximator is less crucial. One intriguing possibility is to learn a nonlinear representation from a set of training instances, then freeze the nonlinear components so that fast least-squares methods can be used on the test instances. For example, STAGE could learn a neural network representation of V^π from training instances, then freeze the input-to-hidden weights of the network to allow linear learning on a new test instance. This could be a useful way to construct a feature set for a linear architecture automatically. (Related ideas are discussed in [Utgoff 96].)

More aggressive optimization of $\tilde{\mathbf{V}}^\pi$. On each iteration, in order to find a promising new starting state for the baseline procedure π, STAGE optimizes $\tilde{V}^\pi$ by performing first-improvement hillclimbing. A more aggressive optimization technique, such as simulated annealing, could instead be applied at that stage; and that may well improve performance.

Steepest descent. With the exception of the WALKSAT results of Section 4.7 and the experiments of Section 5.2.3, STAGE has been trained to predict and improve upon the baseline procedure of π = first-improvement hillclimbing. However, in some optimization problems—particularly, those with relatively few moves available from each state—*steepest-descent* (best-improvement) search may be more effective. Steepest-descent is proper, Markovian, and monotonic, so STAGE applies directly; and it would be interesting to compare its effectiveness with first-improvement hillclimbing's.

Continuous optimization. This dissertation has focused on discrete global optimization problems. However, STAGE applies without modification to continuous global optimization problems (i.e., find $\mathbf{x}^* = \operatorname{argmin} \text{Obj} : \Re^K \rightarrow \Re$) as well. The cartogram design problem of Section 4.6 is an example of such a problem; however, much more sophisticated neighborhood operators than the point perturbations I defined for that domain are available. For example, the downhill simplex method of Nelder and Mead (described in [Press *et al.* 92, §10.4]) provides an effective set of local search moves for continuous optimization. Downhill simplex reaches a local optimum quickly, and Press et al. [92] recommend embedding it within a multiple-restart or simulated-annealing framework. STAGE could provide an effective learning framework for multi-restart simplex search.

Confidence intervals. STAGE identifies good restart points by optimizing $\tilde{V}^\pi(x)$, the predicted expected outcome of search from x. However, in the context of a long run involving many restarts, it may be better to start search from a state with worse expected outcome but higher outcome *variance*. After all, what we really want to minimize is not the outcome of any one trajectory, but the minimum outcome over the whole collection of trajectories STAGE generates. One possible heuristic along these lines would be to exploit confidence intervals on $\tilde{V}^\pi$'s predictions to guide search. For example, STAGE could evaluate the promise of a state x by, instead of the expected value of $\tilde{V}^\pi(x)$, a more *optimistic* measure such as

- the 25th-percentile prediction of $\tilde{V}^\pi(x)$, or
- the probability that $\tilde{V}^\pi(x)$ exceeds the best value seen so far on this run.

Such strategies could have the effect of both encouraging exploration of state-space regions where $\tilde{V}^\pi$ is poorly modeled (similar to the Interval-Estimation [Kaelbling 93] and IEMAX [Moore and Schneider 96] algorithms) and encouraging repeated visits to states that promise to lead occasionally to excellent solutions.

8.2.2 Other Uses of VFA for Optimization

STAGE exploits the value function V^π for the purpose of guiding search to new starting points for π. However, value functions can also aid optimization in at least two further ways: *filtering* and *sampling.*

Filtering refers to the early cutoff of an unpromising search trajectory—before it even reaches a local optimum—to conserve time for additional restarts and better trajectories. Heuristic methods for filtering have been investigated by, e.g., [Nakakuki and Sadeh 94]. Perkins et al. [97] have suggested that reinforcement-learning methods could provide a principled mechanism for deciding when to abort a trajectory. In the context of STAGE, filtering could be implemented simply as follows: cut off any π-trajectory when its predicted eventual outcome $\tilde{V}^\pi(x)$ is worse than, say, the mean of all π-outcomes seen thus far. This technique would allow STAGE to exploit its learned predictions during both stages of search.

Sampling refers to the selection of candidate moves for evaluation during search. In this dissertation, I have assumed that candidate moves are generated with a probability distribution that remains stationary throughout the optimization run. In optimization practice, however, it is often more effective to modify the candidate distribution over the course of the search—for example, to generate large-step candidate moves more frequently early in the search process, and to generate small-step, fine-tuning moves more frequently later in search. Cohn reviews techniques for adapting the sampling distribution of candidate moves, including one "based on their probability of success and on their effect on improving the cost function" [Cohn 92, §2.4.4]. In order to estimate these quantities without having to invoke the (presumably expensive) objective function, Cohn's move generator maintains statistics for each category of move that has been tried recently in the run—a simple kind of reinforcement learning.

A more sophisticated approach has recently been proposed by Su et al. [98]. Their method learns, over multiple simulated-annealing runs, to predict the long-term outcome achieved by starting search at state $\mathbf{x}$ and with initial action a.[1] In reinforcement-learning terminology, their method learns to approximate the task's *state-action value function* $Q^\pi(\mathbf{x}, a)$ [Watkins 89]. This form of value function allows the effects of various actions a to be predicted without having to actually apply the action or invoke the objective function. Their method uses the learned value function to preselect the most promising out of five random candidate moves before each step of simulated annealing, thereby saving time that would have been spent evaluating bad candidate moves. In optimization domains where objective function evaluations are costly, the Q^π value-function formulation offers the potential for significant speedups. It remains for future research to determine how best to combine filtering, sampling, and search-guiding uses of value functions in optimization.

8.2.3 Direct Meta-Optimization

All the approaches discussed in this thesis have built evaluation functions by approximating a value function V^π or V^*, functions which predict the long-term outcomes of a search policy. However, an alternative approach not based on value function approximation, which I call *direct meta-optimization*, also applies. Direct meta-optimization methods assume a fixed parametric form for the evaluation function and optimize those parameters directly with respect to the ultimate objective, sampled by Monte

[1] Note that the state vector $\mathbf{x}$ for simulated annealing consists of both the current configuration x and the current temperature T.

Carlo simulation. In symbols, given an evaluation function $\tilde{V}(x|\vec{w})$ parametrized by weights $\vec{w}$, we seek to learn $\vec{w}$ by directly optimizing the *meta-objective function*

$$M(\vec{w}) = \text{the expected performance of search using evaluation function } \tilde{V}(x|\vec{w})\ .$$

The evaluation functions $\tilde{V}$ learned by such methods are not constrained by the Bellman equations: the values they produce for any given state have no semantic interpretation in terms of long-term predictions. The lack of such constraints means that less information for training the function can be gleaned from a simulation run. The temporal-difference goal of explicitly caching values from lookahead search into the static evaluation function is discarded; only the final costs of completed simulation runs are available. However, there are several reasons to believe that a direct approach may be effective:

- Not having to meet the Bellman constraints may actually make learning easier. For example, even if a domain's true value function is very jagged, meta-optimization may discover a quite different, smooth $\tilde{V}$ that performs well. (This point was also made in [Utgoff and Clouse 91].)
- Direct approaches do not depend on the Markov property. Since they treat the baseline search procedure as a black box, they can be applied to optimize the evaluation function for backtracking search algorithms such as A*, or in sequential decision problems involving hidden state.
- Similarly, extra algorithmic parameters such as hillclimbing's *patience* level and simulated annealing's temperature schedule can be included along with the evaluation function coefficients in the meta-optimization.

Further arguments supporting the direct approach are given in [Moriarty *et al.* 97].

Direct meta-optimization methods have been applied to learning evaluation functions before, particularly in the game-playing literature. Genetic-algorithm approaches to game learning generally fall into this category (e.g., [Tunstall-Pedoe 91]). Recently, Pollack et al. attacked backgammon by hillclimbing over the 3980 weights of a neural network [Pollack *et al.* 96]. Surprisingly, this procedure developed a good backgammon player, though not on the level of Tesauro's TD-Gammon networks. Meta-optimization has also been applied successfully to aid combinatorial optimization. Ochotta's simulated annealing system for synthesizing analog circuit cells made use of a sophisticated cost function parametrized by 46 real numbers [Ochotta 94]. These and 10 other parameters of the annealer were optimized using Powell's method as described in [Press *et al.* 92]. Each parameter setting was evaluated by summing the mean, median and minimum final results of 200 annealing runs on a small representative problem instance. After several months of real time and four years of CPU time (!), Powell's method produced an evaluation function which performed well and generalized robustly to larger instances.

I believe that the computational requirements of direct meta-optimization can be significantly reduced by the use of new memory-based stochastic optimization techniques [Moore and Schneider 96, Moore *et al.* 98]. These techniques are designed to optimize functions for which samples are both expensive to gather and potentially very noisy. The meta-objective function M certainly fits this characterization, since sampling M means performing a complete run of the baseline stochastic search procedure and reporting the final result. An important future direction for reinforcement-learning research is to carefully compare the empirical performance of direct meta-optimization and value function approximation methods.

8.3 Concluding Remarks

In the decade since the deep connection between AI heuristic search and Markov decision process theory was first identified, the field of reinforcement learning has made much progress. Algorithms for learning sequential decision-making from simulation data are now well understood for tasks in which the value function can be represented exactly. This thesis has addressed the more difficult case in which the value function must be represented compactly by a function approximator. Its primary contribution is a practical algorithm, built on reinforcement-learning foundations, that efficiently learns and exploits a predictive model of a search procedure's performance. Ultimately, I believe such methods will lead to more effective solutions to large sequential decision problems in industry, science, and government, and thereby improve society.