Analysis of STAGE Algorithm based on Solving Bin Packing Problem [*]

Gholamreza Haffari
Computer Engineering Department
Sharif University of Technology
Tehran, Iran
P.O.Box 13445-1669

Saeed Bagheri Shouraki
Computer Engineering Deaprtment
Sharif University of Technology
Tehran, Iran
P.O.Box 13445-1669

Abstract

Previous researches have shown the success of using Reinforcement Learning in solving combinatorial optimization problems. The main idea of these methods is to learn (near) optimal evaluation functions to improve local searches and find (near) optimal solutions. STAGE algorithm, introduced by Boyan & Moore, is one of the most important algorithms in this area. In this paper, we focus on Bin-Packing problem, an important NP-Complete problem. We analyze cost surface structure of this problem and investigate "big valley" structure for the set of its local minima. The result gives reasons for STAGE's success in solving this problem. Then based on experimental results of Bin-Packing problem, we analyze the effectiveness of using different local search algorithms and different learning structures in STAGE.

Keywords: Combinatorial optimization, Reinforcement Learning, STAGE algorithm.

1 Introduction

Large scale optimization problems are in great importance in all fields of science, engineering and operation research. The goal of each of these problems is to find the best possible configuration from a large space of possible configurations. Unfortunately, most of these problems are NP-Hard [9], and finding their optimum solution in reasonable amount of time is almost impossible. Thus, there has been a great deal of work on heuristic methods for finding approximate solution in limited amount of time. Some of these methods, called Approximation Algorithms [12], are based on strong theoretical background that guarantees the quality of approximate solution in the specific distance of optima. For example, many approximation algorithms have been proposed and analyzed for Bin-Packing problem in [8]. But these algorithms are special-purpose, i.e., they are specific to particular problems, so general-purpose heuristic methods are emerged.

General-purpose heuristic methods do not guarantee the quality of solutions in the way that Approximation Algorithms do, but practically they find good solutions. Frequently, these heuristic search methods, such as Simulated Annealing and Genetic Algorithm, are based on iteration and are easy to implement. Algorithms, which use Reinforcement Learning methods in solving combinatorial optimization problems, are in this category.

By adopting the familiar state-space search perspective to a combinatorial optimization problem [5], a good solution is found by starting in some initial state and applying Greedy-Descent policy (usually based on an evaluation function) to eventually reach some final good state. With this viewpoint, Reinforcement Learning methods are used to learn an evaluation function that predicts the outcome of the local search, and to guide search to low-cost solutions using this learned evaluation function. Note that the evaluation function is not limited to the same form as the objective function. In addition to providing a good measure for the features of a state (directly related to the objective function), an evaluation function also gives some hints on which states predictably lead to good states using some local search algorithm [1]. Based on this idea, Zhang and Dietterich applied Reinforcement Learning to the Space Shuttle Payload Processing domain ([6], [10]), and Boyan and Moore introduced STAGE algorithm that has shown excellent performance on a wide range of optimization problems [2]. By combining aspects of these two works, Reinforcement Learning was used in solving the Dial-A-Ride problem, a compli-

[*]accepted to The International Conference on Machine Learning and Applications (ICMLA'01), Las Vegas, USA.

cated variant of TSP [4].

In this paper we focus on Bin-Packing problem and investigate some interesting characteristics of STAGE in more details. First of all, we analyze cost surface structure of the Bin-Packing problem and examine the "big valley" structure for the set of its local minima. The result confirms previous works done for TSP and graph-bisection problem that the cost surfaces exhibit globally convex structure [7]. It gives further insight to the success of STAGE in solving Bin-Packing problem. Then, we investigate some interesting characteristics of STAGE based on the results of experiments on Bin-Packing problem. We compare the effectiveness of steepest-descent hill climbing, stochastic hill climbing, and first-improvement hill climbing as the local search algorithms in STAGE. We also examine the effect of using different learning structures on STAGE's performance. In particular, we use logarithmic function, exponential function, and CMAC network as evaluation function approximators, and compare their results to that of quadratic polynomial.

The rest of the paper is organized as follows. First, it gives background information and reviews the definition of Bin-Packing problem and STAGE algorithm in section 2. Analyzing cost surface structure of this problem comes in section 3. The effect of changing local search algorithm in STAGE comes in section 4. Analyzing the effect of different function approximators comes in section 5. Finally, we outline conclusions and future works.

2 Background

2.1 Bin-Packing Problem

Bin-Pacing is a classical NP-Complete problem [9]. This problem has many real-world applications, including loading trucks subject to weight limitations, packing commercials into station breaks, and cutting stock materials from standard lengths of cable or lumber [8].

In this problem, we are given a bin capacity C and a list $L = (a_1, a_2, \ldots, a_n)$ of n items, each having a size $s(a_i) > 0$. The goal is to pack the items into as few bins as possible, i.e. partition them into a minimum number m of subsets $B_1, B_2, \ldots, B_m$ such that for each $B_j : \sum_{a_i \in B_j} s(a_i) < C$.

To view this problem as a state-space search problem, we need the definition of state and neighborhood structure. A solution state x simply assigns a bin number $b(a_i)$ to each item. Each item is initially placed alone in a bin: $b(a_1) = 1, b(a_2) = 2, \ldots, b(a_n) = n$. Neighboring states can be generated by moving any single item a_i into a random other bin with enough spare capacity to accommodate it [1].

2.2 STAGE Algorithm

The central idea of STAGE is learning to predict which starting state is more promising for some local search algorithm, from sample search trajectories ([2], [1]). In addition to the usual objective function, STAGE creates and tries to learn a new evaluation function for predicting how promising a state is as a starting point for some search algorithm. The new evaluation function is approximated with some form of function approximation such as polynomial regression or multilayer perceptron. Each state is represented as a real-valued feature vector, where the relevant features are handpicked in advance. STAGE repeatedly alternates between two stages of local search: running the original search algorithm on the objective function and running hill climbing on the new evaluation function to find a promising new starting state for the original search algorithm. In each iteration, STAGE tries to learn the new evaluation function from the available search trajectories. The training data can be obtained by Monte-Carlo simulation.

To guarantee convergence, STAGE requires the search algorithm to be proper (terminates with probability one) and behaves as a Markov chain [1]. When the search algorithm satisfies Markov property, all intermediate states on each simulated trajectory can be considered as alternate starting points for that search, thus to obtain many training data from a single search trajectory. When a local minimum for both the original objective function and the new evaluation function is reached, STAGE resets search to a random starting point.

3 Cost Surface Structure of Bin-Packing Problem

Recent analyses of cost surface of optimization problems show that as problems grow large, random local minima are almost surely of "average" quality, and "central limit catastrophe" happens for them; consequently all but an exponentially small number of local minima will have a cost approximately equal to that of the average local minimum [3]. If we exploit a structure for the cost surface of the problem, the best previously found local minima can be used to intelligently suggest next starting points for a greedy-descent algorithm, in such a way that lead to lower-cost solutions.

The method, which we use to study cost surface structure for Bin-Packing problem, is similar to that of Boese and others for exploiting cost surface structure for TSP and the graph-bisection problem [7]. They examined the set of local minima from the perspective of the best local minimum. As we see later in this section, the experimental results confirm the "globally convex" structure for Bin-Packing problem.

For exploiting a structure for the cost surface of Bin-Packing problem, we need to define a neighborhood structure, and equivalently an operator. As we can see from section 2.1, this operator moves one item from a bin to another bin, which has enough free space to accommodate it. We define the distance between two solutions x_1 and x_2 as the minimum number of operators needed to transform x_1 to x_2 and denote it by $d(x_1, x_2)$. Since computation of $d(x_1, x_2)$ is time consuming, we measure the similarity between x_1 and x_2 according to the number of items that are in the same bins in both solutions. We will use the term bond distance, denoted $b(x_1, x_2)$, equal to $2 * number\_of\_bins$ minus similarity between x_1 and x_2, and use it as the estimation for $d(x_1, x_2)$.

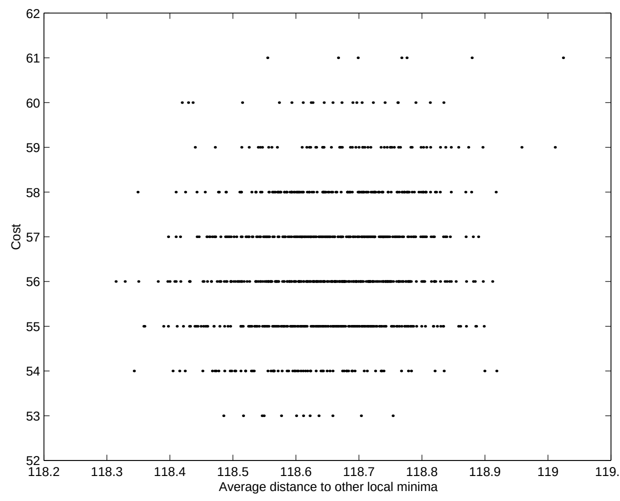

Figure 1: Analysis of 4001 random locally minimum solutions for bin-packing instance problem u120_00, the data represent 4001 distinct local minimum. This figure plots average distance to other local minima.

To find how local minima correlate with each other, we obtain 4001 random locally minimum solutions for $u120\_00$ instance of Bin-Packing problem[1], which has 120 bins of capacity 150. Figure 1 plots the solution cost versus its average bond distance to all (4000) other local minima, and figure 2 shows the solution cost versus its distance to the best local minimum. A "random" local minimum is found by starting at a random

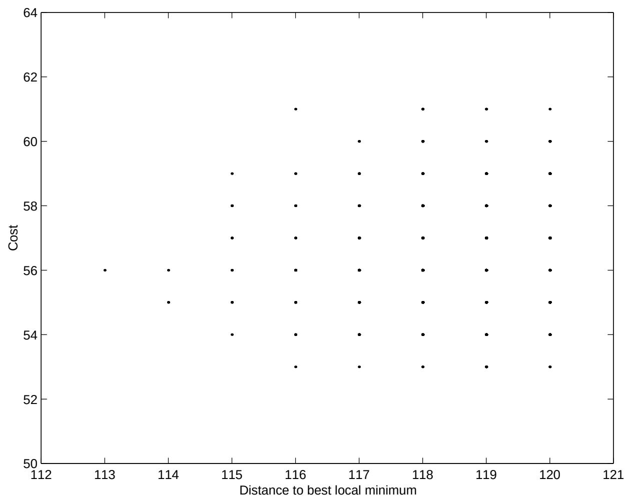

Figure 2: Analysis of 4001 random locally minimum solutions for bin-packing instance problem u120_00, the data represent 4001 distinct local minimum. This figure plots distance to the best local minimum.

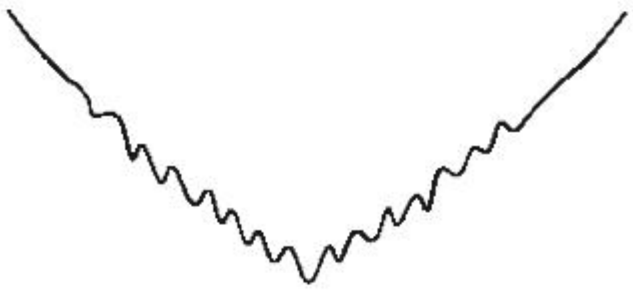

Figure 3: Intuitive picture of the "big valley" search space structure [3].

initial solution and executing a greedy-descent algorithm. In the figure 2 we can see a clear correlation: the best local minimum appears to be "central" to all other local minima, and indeed a "big valley" structure can be said to govern the set of locally minimum solutions, as illustrated in figure 3.

In Bin-Packing problem, the objective function for STAGE to minimize is the number of bins used. For automatic learning of its secondary evaluation function, Boyan provided STAGE with two state features [1]: the number of bins used, and the variance in bin fullness level. STAGE learned its evaluation function by quadratic regression over these two features. By choosing quadratic regression for evaluation function, STAGE implicitly exploited the cost surface structure for the Bin-Packing problem. To show

[1]All instances of Bin-Packing problem are from Operation Research Library. For further information see this web page: http://www.ms.ic.ac.uk/info.html.

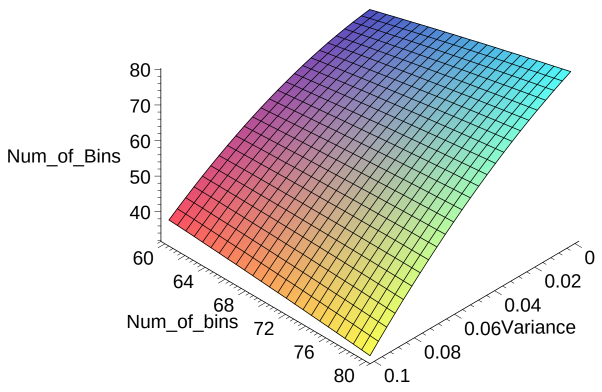

Figure 4: A typical learned evaluation function for u250_00 instance problem.

the relationship between cost surface and evaluation function, figure 4 shows a typical evaluation function learned by STAGE for $u250\_00$ instance problem.

The number of neighborhood states grows rapidly as the number of bins increases; so stochastic hill climbing has been used by Boyan as the local search algorithm in STAGE for optimizing evaluation function. In the next section, we want to see the effect of other local search algorithms on STAGE's performance.

4 Different Local Search Algorithms in STAGE

In this section, based on the experimental results from Bin-Packing problem, we compare the effects of different local search algorithms, including stochastic hill climbing, first-improvement hill climbing, and steepest-descent hill climbing, on STAGE's performance.

Steepest-descent hill climbing takes a step from a state to one of its neighbor states that maximally improves objective function. For search problems where number of neighbors of a state is huge, stochastic hill climbing is cheaper to run than steepest-descent hill climbing. Stochastic hill climbing with no equal-cost move considers limited numbers of neighbors of a state randomly, and takes one of them which enhances the objective function[2] First-improvement hill climbing systematically examines all of the neighbors and selects the first state, which is better than the current state. If no neighbor improves objective function, the search trajectory terminates. All of these algorithms are

[2]In stochastic hill climbing, we cut off the search process when Patience consecutive moves produce no improvement.

Table 1: Summary of STAGE local search algorithms.

ALGORITHM	DESCRIPTION
FIHC	FIRST-IMPROVEMENT HILL CLIMBING
SDHC	STEEPEST-DESCENT HILL CLIMBING
STHC	STOCHASTIC HILL CLIMBING PATIENCE = 250.

Table 2: Experimental results of solving different instances of Bin-Packing by STAGE with different local search algorithms for problem instances with 500 bins of capacity 150.

INSTANCE	ALG.	MEAN	BEST	WORST
U500_00	FIHC	208.80 ±0.472	207	211
	SDHC	214.50 ±3.021	209	244
	STHC	212.80 ±0.859	209	216
U500_01	FIHC	211.80 ±0.505	210	215
	SDHC	219.64 ±5.402	211	260
	STHC	215.50 ±0.735	211	218
U500_02	FIHC	212.00 ±0.439	211	215
	SDHC	223.05 ±6.475	212	260
	STHC	216.10 ±0.900	210	219
U500_03	FIHC	214.89 ±0.439	212	216
	SDHC	225.50 ±7.371	215	273
	STHC	218.35 ±0.518	215	220
U500_04	FIHC	216.05 ±0.670	213	221
	SDHC	231.00 ±8.251	216	271
	STHC	220.10 ±0.637	218	224

strictly monotonic, Markovian, and (if the search space is finite) proper. Table 1 shows the summary of these algorithms.

In our experiments, each instance has 500 bins of capacity 150. STAGE is limited to 500000 total moves. The results of 30 runs of STAGE with each algorithm for each instance are summarized in Table 2, each line reports the mean, 90% confidence interval[3], best, and worst solutions found by 30 independent runs. The effect of each local search algorithm on $u500\_00$ and $u250\_00$ instance problems is displayed in figure 5 and figure 6.

As it can be concluded from experimental results of Table 2, FIHC outperforms other local search algorithms with respect to the value of mean, best, and worst solutions that it has found. The reason for its good results is that it explores the search space as it could as possible, i.e., it tries paths about which little is yet known, while it exploits a search branch, i.e., it pursues what appears to be the best path given the limited observations made thus far. After FIHC, STHC has shown good effectiveness. This algorithm visits more areas of the search space than FIHC do but uses little informa-

[3]The confidence interval for the mean is produced by $\mu \pm t_{\frac{\alpha}{2},N-1}\frac{S}{\sqrt{N}}$,where $1-\alpha$ is the confidence factor, N is the number of runs, and t is Student_t distribution.

tion from previously visited states. Finally, SDHC has produced the worst results because it strongly sticks to a search branch to produce a local minimum solution and needs much more time to explore other areas of search space. It also can be seen from figure 5 and figure 6 that STHC and FIHC have similar effectiveness and oscillate much more than SDHC. The reason is that they explore search space far more than SDHC do, so they find many more local minima than SDHC.

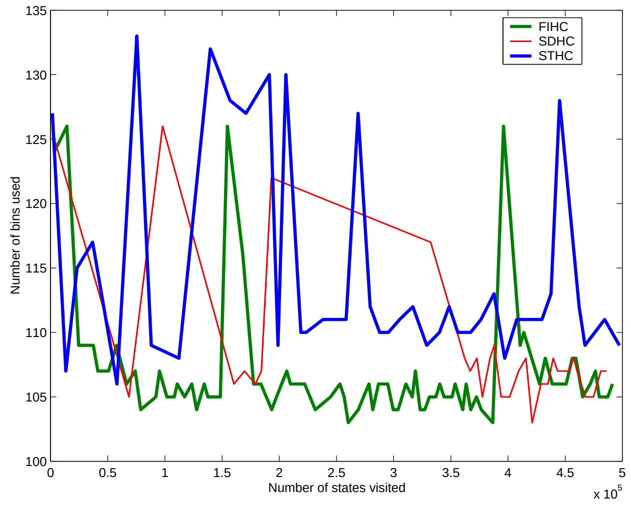

Figure 5: Effect of using different local search algorithms on STAGE's performance. This figure plots the results for u250_00 instance problem.

5 The effect of using different function approximators

An important parameter to STAGE is the fitter or function approximator used to model the value function; STAGE relies on this function to predict the eventual outcome of the base local search algorithm from an unvisited state. In this section, we analyze the effect of using different function approximators such as exponential function, logarithmic function, and CMAC network on STAGE performance and compare their results to that of quadratic polynomial.

Our intuition of cost structure of combinatorial optimization problems gives us some hints for choosing the proper fitter. We have used the exponential function to exploit the "big-valley" structure of local minima:

$$g(\mathbf{x}) = ke^{\frac{\|\mathbf{x}-\mathbf{m}\|^2}{2\sigma^2}} \tag{1}$$

where k and σ are scalar real numbers, and $\mathbf{m}$ is the center vector. We pose a least square problem, i.e. k, σ and $\mathbf{m}$ must be determined in such a way that

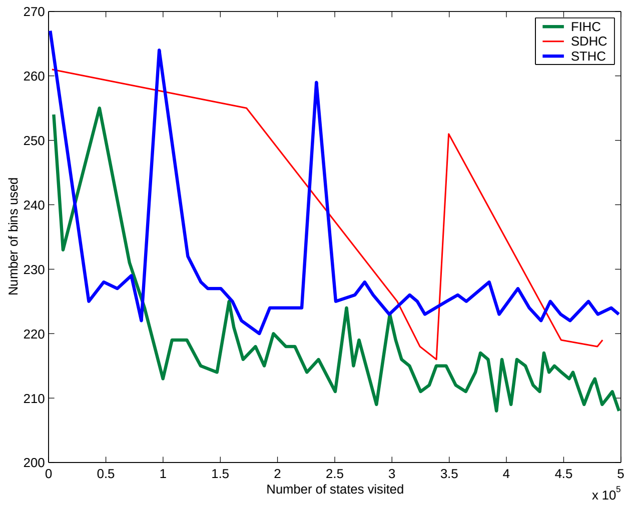

Figure 6: Effect of using different local search algorithms on STAGE's performance. This figure plots the results for u500_00 instance problem.

minimizes:

$$\sum_i (y_i - g(\mathbf{x}_i))^2 \tag{2}$$

Equivalently, we transform the original problem to the problem of fitting quadratic polynomial on $\log(y)$. We have also considered the logarithmic function as the fitter:

$$l(\mathbf{x}) = \log\left(\frac{\|\mathbf{x}-\mathbf{m}\|^2}{2\sigma^2} + k\right) \tag{3}$$

Where k and σ are scalar real numbers, and $\mathbf{m}$ is a vector; they must be determined in such a way that the error be minimized. Again, we consider the problem of fitting quadratic polynomial on e^y.

Boyan has mentioned qualities of the fitter which make it suitable for STAGE [1]; it must be incremental, noise-tolerant, and producing good values for unvisited new states. With these considerations, he has suggested the use of linear architecture approximators. We have used CMAC[4] network as the linear architecture approximator. The operation of these networks can be defined in terms of large set of overlapping, multi-dimensional receptive fields with finite boundaries [11]. Any input vector falls within the range of some of the receptive fields and falls outside many other receptive fields. The response of the network is the weighted sum of the responses of the receptive fields excited by the input, and is not affected by other receptive fields; the distance between input vector and receptive fields determines the weights. Sim-

[4]Cerebellar Model Arithmetic Computer

Table 3: Experimental results of solving different instances of Bin-Packing by STAGE with different function approximators for problem instances with 500 bins of capacity 150. Each line reports the mean, 90% confidence interval , best, and worst solutions found by 30 independent runs.

Instance	Fitter	Mean	Best	Worst
u500_00	Exp	213.44 ±3.61	208	251
	Log	235.94 ±6.12	212	251
	Cmac	218.94 ±0.64	215	222
	Poly	210.39 ±0.83	206	214
u500_01	Exp	214.69 ±0.93	210	218
	Log	244.80 ±4.37	217	255
	Cmac	222.60 ±0.50	220	225
	Poly	214.10 ±0.94	210	220
u500_02	Exp	217.50 ±3.73	210	255
	Log	244.30 ±4.43	218	255
	Cmac	223.14 ±0.56	220	225
	Poly	216.39 ±2.11	211	236
u500_03	Exp	217.14 ±0.87	212	221
	Log	244.60 ±5.33	220	258
	Cmac	225.55 ±0.71	222	229
	Poly	217.85 ±1.15	213	225
u500_04	Exp	219.39 ±0.84	215	222
	Log	244.69 ±4.80	227	260
	Cmac	227.30 ±0.51	224	229
	Poly	218.94 ±0.89	216	223

ilarly, training for a given input vector only affects adjustable parameters of the exited receptive fields.

We have obtained our experimental results by applying STAGE on problem instances with 500 bins, the results are summerized in Table 3. STAGE is limited to 150000 moves, so we can compare the effect of allowing more moves in STAGE (compare the results for polynomial approximator in Table 2 and Table 3). The more moves we allow in STAGE, the more accurate the final solution will be.

As it can be seen in Table 3, polynomial regression outperforms other fitters, i.e. it finds better solutions than exponential function, logarithmic function, and CMAC network. However, exponential fitter is very similar to polynomial fitter with respect to the quality of final solutions. We have also noticed that when STAGE is allowed to do small number of moves (nearly 50000 moves), exponential approximator outperforms polynomial regression. We relate this phenomenon to the distribution of local minima in the solution space; local optima reside on exponential like function in the solution space of the problem.

Another important issue is related to CMAC network. It finds worse result than that of polynomial and exponential function approximators, but recall that we do not give any pre-knowledge about cost structure to this approximator. In suggesting quadratic polynomial and exponential approximators, we implicitly have used the results of section 3, that the "big valley" structure governs local minima of the problem, so we chose functions that have similar shape to this structure to exploit this knowledge. In contrast, CMAC network tries to construct this relationship among local minima and then uses it to guide search process, i.e. it automatically finds the structure that governs local minima.

6 Conclusions and future works

Recent researches have shown the success of using Reinforcement Learning in solving combinatorial optimization problems. In this paper, STAGE algorithm, which is one of the most important algorithms based on Reinforcement Learning, was analyzed from the experimental results on Bin-Packing problem. We studied the cost surface structure for the Bin-Packing problem. It was illustrated that "big valley" structure governs its set of local minima which gives us further reasons why STAGE has produced good results for this problem. We examined different local search algorithms for STAGE and compared their relative effectiveness: first-improvement hill climbing outperforms stochastic hill climbing and steepest-descent hill climbing in solving Bin-Packing problem, where the number of neighboring states is nearly large.

We also analyzed the effect of using logarithmic function, exponential function, and CMAC network as the learning structure on STAGE's performance. It is shown that polynomial and exponential function approximators have nearly identical performance, and outperform other approximators. However, it is worth noting that CMAC network do not use any pre-knowledge about the shape of cost structure and learns it automatically. Further research on STAGE's performance will investigate using other function approximators such as multi-layer perceptron and radial basis functions for estimating evaluation function. It is also interesting to analyze STAGE's performance when previously learned evaluation functions are used for solving new instances of a problem.

Acknowledgements

The first author is very thankful to John N. Tsitsiklis for sending his useful book, Andrew W. Moore for his useful information about STAGE, and Gholamreza Setudeh for useful discussions on statistical analysis of the experimental results. We also would like to acknowledge Mansour Jamzad, Mohammad Abam and Zahra Zahedi Kermani for their research facilities.

References

[1] J. A. Boyan, Learning Evaluation Function for Global Optimization, Doctoral dissertation, Computer Science Department, Carnige Mellon University, 1998.

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

[3] K. D. Boese, Models for Iterative Global Optimization, Doctoral dissertation, Computer Science Department, University of California, Los Angeles, 1996.

[4] R. Moll, A. G. Barto, T. J. Perkins, and R. S. Sutton, Learning Instance-Independent Value Functions to Enhance Local Search, Proceeding of NIPS-98, Denver, 1998.

[5] D. P. Bertsekas and J. N. Tsitsiklis, Neuro-Dynamic Programming, Athena Scientific, Belmont, MA, 1996.

[6] W. Zhang and T. G. Dietterich, A reinforcement learning approach to job-shop scheduling, In Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI) pp. 1114–1120, 1995.

[7] K. D. Boese, A. B. Khang, and S. Muddu, On the Big Valley and Adaptive Multi-Start for Discrete Global Optimization, Operation Research Letters 16(2), 1994.

[8] E. G. Coffman, M. R. Garey, and D. S. Johnson, Approximation algorithms for bin packing: a survey, In D. Hochbaum, editor, Approximation Algorithms for NP-Hard Problems, PWS Publishing, 1996.

[9] M. R. Garey and D. S. Johnson, Computer and Intractability: A Guide to the Theory of NP-Completeness, W. H. Freeman, 1979.

[10] W. Zhang, Reinforcement Learning for Job-Shop Scheduling, Doctoral dissertation, Computer Science Department, Oregon State University, 1996.

[11] W. T. Miller and F. H. Glanz, UNH_CMAC Version 2.1: The University of New Hampshire Implementation of the CMAC, http://www.ece.unh.edu/robots/unh_cmac.ps, 1996.

[12] Vijay V. Vazirani, Approximation Algorithms, Springer-Verlag, New York, 1999.