New Approaches to Design and Control of Time Limited Search Algorithms

Partha Pratim Chakrabarti[1] and Sandip Aine[2]

[1] Dept of CSE, Indian Institute of Technology Kharagpur, 721302, India
ppchak@cse.iitkgp.ernet.in
[2] Mentor Graphics (India) Pvt. Ltd., Noida, UP 201301, India
sandip_aine@mentor.com

Abstract. We talk about two key aspects of the quality-time trade-offs in time limited search based reasoning namely, design of efficient anytime algorithms and formulations for meta-reasoning (or control) to optimize the computational trade-off under various constrained environments. We present the ideas behind novel anytime heuristic search algorithms, both contract and interruptible. We also describe new meta-control strategies that address parameter control along with time deliberation.

1 Introduction

Optimizing the quality-time trade-off while solving optimization problems has been a major area of interest for Artificial Intelligence (AI) researchers. Since many of the optimization problems encountered in practice such as design, planning and scheduling are *NP-hard*, optimal rationality is not to be expected within limited resources (like computational time). In AI literature, this trade-off is addressed in two phases, through *design* of efficient algorithms which provide handles to control their execution, and through intelligent *meta-control* mechanisms which automatically decide such execution strategies under various constrained scenarios. The use of *Anytime algorithms* [1] was proposed as a basic computational framework for handling the quality-time trade-offs. These algorithms work under *any time* limitations producing different quality solutions, and thus, provide opportunities to reason about their time deliberation. Anytime algorithms are classified into two categories namely, interruptible algorithms and contract algorithms. Interruptible anytime algorithms are expected to regularly produce solutions of improved quality and when suddenly terminated, return the best solution produced so far as the result. For a contract algorithm, time is allocated a priori and the algorithm is required to provide a high-quality solution at the end of the time contract. Using these anytime algorithms as the base level computational model, several *meta-reasoning* frameworks have been proposed to control the quality-time trade-offs for hard optimization problems.

This paper addresses two dimensions of the quality-time trade-off paradigm, namely design and meta-control [2]. In the design domain, we concentrate on Anytime Heuristic Search techniques and present the key ideas behind two novel

S. Chaudhury et al. (Eds.): PReMI 2009, LNCS 5909, pp. 1–6, 2009.

Anytime Heuristic Search algorithms, suitable for contract and interruptible purposes. In the control domain, we introduce the concept of parameter control of anytime algorithms and propose integrated meta-level frameworks capable of taking a unified decision on the parameter configuration chosen along with the deliberation control. We focus on the formulations and strategies required for an integrated control of iterative stochastic optimization algorithms such as Simulated Annealing [3] or Evolutionary Algorithms [4] under a variety of scenarios.

2 Design of Anytime Heuristic Search Techniques

In the past few years, a number of interruptible heuristic search algorithms have been developed. In general, these algorithms follow two basic principles: they use depth guided non-admissible pruning to obtain fast solutions and work in multiple iterations with gradual relaxation of the constraints to produce a stream of gradually improving solutions. These algorithms can be broadly classified into two categories, weighted A* [5,6] approaches and beam based approaches [7,8]. One of the major disadvantages of using the anytime techniques is the lack of models, which can provide the user an apriori estimate about the quality-time trade-off. Therefore, to use these techniques, lengthy simulations are required to find appropriate parameter settings. In the next sub-sections we discuss two novel techniques of heuristic search. The first one is designed to work under node limitation contract (Contract Search) and the second one is an interruptible algorithm (Anytime Window A*).

2.1 Contract Search

We explore the possibility of developing a heuristic search algorithm under a contract. Since time spent can be determined in terms of number of nodes expanded (as node expansion is the heaviest task for a search algorithm), we work under a node expansion contract. The A* algorithm performs a global competition performed among nodes in the open list, select nodes in ascending order of their $f(n)$ values. While this global competition guarantees optimality of solutions produced by the algorithm (when heuristics are admissible), it also ensures that all nodes which come into the open list and have $f(n)$ less than the optimal cost solution are necessarily explored. Our work is based on the hypothesis that for each level of a search space, if we only expand nodes up to the 'optimal-path node', we can attain optimality. To formalize this we introduce the concept of ranking among nodes in the search space. The rank denotes the position of a node (in terms of f-value) among a chosen set of nodes. While A* uses a global ranking scheme (without any restrictions) we introduce a local ranking among the nodes in the same level of a search. Our study of state spaces of some important practical problems reveals that the nodes on the optimal path are usually very competitive among nodes at their same level. We reinforce this observation by obtaining analytical results on search tree models (uniform and non-uniform costs).

Our analysis on best-first search shows that stricter pruning can be obtained using *rank based restrictions*. For Contract Search, we propose to obtain a probabilistic estimation of rank values, and use that information to guide the search. For this, we put forward the concept of Probabilistic Rank Profile (PRP). Probabilistic Rank Profile $P(S|l, k(l))$ is a model which represents the chance of expanding the 'optimal-path' node at a level if a maximum number of $k(l)$ nodes are expanded at that level. These values can be obtained either through profiling or by using search space characteristics to generate the PRP values. For Contract Search, we use the PRP to compute the expansion bounds for each level of the search space depending on the contract specification. We formulate the bound selection (k-selection) problem under a given contract C as follows: For a problem having PRP $P(S|l, k(l))$ we generate a k-cutoff for each level l $(0 \leq l \leq h)$ such that,

$$\sum_l k(l) \leq C \; and \;\; P_S(C) \text{ is maximized} \tag{1}$$

With this formulation, we define the k-selection strategy for a particular level as a Markov Decision Process (MDP) which is solved using dynamic programming. The strategy can be computed off-line and then used to guide the actual search under specified node limitations.

Contract Search uses these $k(l)$ values to limit the expansions for a particular level. A naive way of incorporating the restrictions would be to expand the best $k(l)$ nodes at level l. In Contract Search, nodes are expanded in the best first mode across all levels. However, if the number of nodes expanded at a given level l equals the $k(l)$ value, the level is suspended. The search terminates when all levels are suspended or there are no more nodes having $f(n)$ less than the obtained solution. It may be noted that when a given level l is suspended, the suspension is propagated to upper levels to reduce useless expansions. Contract Search has been applied on a number of search problems namely, Traveling Salesperson Problem (TSP), 0/1 Knapsack Problem, 15-Puzzle Problem and Instruction Scheduling Problem and have yielded better quality results as compared to schemes such as ARA* [6] and beam search [8].

2.2 Anytime Window A*

As discussed earlier, the algorithm A* considers each node to be equivalent in terms of information content and performs a global competition among all the partially explored paths. In practice, the heuristic errors are usually distance dependent [9]. Therefore, the nodes lying in the same locality are expected to have comparable errors, where as the error may vary substantially for nodes which are distant from each other. Thus, if the global competition performed by A* is localized within some boundaries, tighter pruning can be obtained. Based on this observation, we present an alternative anytime heuristic search algorithm Anytime Window A* (AWA*), which localizes the global competition performed by A* within a fixed-size window comprising of levels of the search tree/graph.

The AWA* algorithm works in two phases, in the inner loop it uses the Window A* routine with a fixed window size and computes the solution under the

current restriction. In the outer loop the window restrictions are gradually relaxed to obtain iteratively improving solutions. The Window A* routine works with a pre-specified window size. For a given window size ω, the expansions are localized in the following way. When the first node of any level (say l) is expanded, all nodes in the open list which are from level $(l - \omega)$ or less will be suspended for this iteration. The window slides in a depth-first manner when deeper nodes are explored. Window A* terminates when the first goal node is expanded or if there is no chance to obtain a better solution (than previously generate). The AWA* algorithm calls the Window A* routine multiple times with gradual relaxation of the window bounds. At the start, window size is set to 0 (depth-first mode). Once the Window A* routine terminates, AWA* checks whether there are any nodes suspended in the previous iteration. If the suspended list is empty, the algorithm is terminated returning the optimal solution. Otherwise, the window size is increased in a pre-decided manner, and Window A* is called again.

We analyze the characteristics of the presented algorithm using a uniform search tree model. This reveals some very interesting properties of the algorithm in terms of heuristic accuracy versus solution quality and related search complexity. The experimental results obtained on TSP and 0/1 Knapsack corroborate the analytical observations, thus, validating the applicability of analysis in real life domains. We get considerable improvement in convergence probabilities as well as intermediate solution qualities over the existing approaches.

3 Integrated Frameworks for Meta-level Control

In the domain of meta-level frameworks, most of the suggested methodologies attempt to solve the problem of time (resource) deliberation only. However, for many algorithms targeted to solve *hard* problems, time allocation is not the only criterion that can influence the solution quality. The progress of these algorithms is strongly affected by some other parameter choices. Thus, an integrated framework for deliberation and control is necessary to effectively handle constrained scenarios. Also, many of the available utility based formulations are for single problem instances. On the other hand, in many real world situations, the requirement is usually to solve single/multiple problem instances under time/quality constraints. In this section, we introduce the concept of parameter control of anytime algorithms under various constrained environments. We mainly concentrate on two specific stochastic techniques namely, simulated annealing (SA) [3] and evolutionary algorithms(EA) [4].

The basic problem for the meta-level controller that we address in this part is to decide on a control parameter setting (along with the time allocation for utility based framework) for anytime algorithms which optimizes the objective function chosen under the specified constraints. For this, the meta-level control framework requires a model to measure the improvement of the solution quality with time for different control settings. Dean & Boddy [10] introduced the term *performance profile* of an algorithm, which for an interruptible algorithm, represents the expected solution quality as a function of the allocated time. This

model suggested in [10] was suitable for static reasoning only. Hansen and Zilberstein [11] extended the static profiling concept for dynamic decision making.

We shift our focus from the time adjustment frameworks to a more generic multi-dimensional control space, the quality-time models no longer remain sufficient. We require to model an algorithm's progress not only with time but also with the different parameter settings. We introduce the concept of Dynamic Parameterized Performance Profile (DPPP) The dynamic parameterized performance Profile of an algorithm, $P(S_j|S_i, \Delta t, C)$, is defined as the probability of reaching a solution state S_j if an additional time Δt is given and the control parameter/vector is C, when the current solution state is S_i. Using this we generate the profiles for SA (with temperature as the parameter) and EA (with mutation and crossover rates as parameters).

Using these profiles we generate the optimal strategies for parameter control of an anytime algorithms for the runtime constrained scenario. The optimal selection strategy can be formulated as:

$$EQ_D(S_i, T_{left}) = \max_{\Delta t, C} \begin{cases} \Sigma_j P(S_j|S_i, T_{left}, C) * Q(S_j) & \text{If } \Delta t = T_{left} \\ \Sigma_j P(S_j|S_i, \Delta t, C) * EQ_D(S_j, T_{left} - \Delta t) - M_p & \\ \text{Otherwise} & \end{cases} \tag{2}$$

The probabilities are obtained from the DPPP, S_i is the current solution state, T_{left} is the time left before reaching the deadline and $Q(S_j)$ is the quality component of the solution state S_j. M_p is the quality penalty for each monitoring step.

The strategies obtained can also be extended for a *utility* based system. Utility is a function of the solution quality and the computation effort (time) which represents the intended trade-odd between quality and time. For a utility based system the framework takes a combined decision on the time allocation and control parameters following the same dynamic programming based approach. In many real-world cases, it may be required to solve a number of problem instances under given run-time limitations. The individual problem instances may depend on each other or may be independent. In our formulation with multiple independent problems, the number of problems to be solved and the total time allocated (i.e., the global deadline) are known a priori. With this formulation, the objective of the meta-controller is to maximize the overall quality within the global deadline. Considering dynamic strategies for multiple problems, we have two options namely, preemptive and non-preemptive strategies. In the non-preemptive case, the problem instances are solved in a particular order and once a problem is evicted from execution queue it is not taken back. In the case of preemptive strategy, any problem instance can be chosen from the problem set at run-time, irrespective of whether it has been stopped earlier or not and the parameter configuration and the time allocation are decided accordingly. Similar control strategies can be generated for the preemptive case. For multiple dependent problems, we consider a pipeline like structure to model an optimization flow, where output obtained from a given stage is fed to the next stage as input.

In this formulation, the controller has an n-stage pipeline where each stage is an optimization algorithm, the total allocated time (or the deadline) D is known a priori. The controller decides on the time allocation and parameter configuration for each stage, such that the expected output quality is maximized. To generate the meta-level strategies for the pipeline structure we modify our profiling technique. Up to this stage, we have generated independent profiles for the individual problems. However, when we consider the optimization process as a multi-stage pipeline, the performance of a given stage will be dependent on the solution obtained from its previous stage. Thus, we condition the profiles by including of information about the input.

Experiments performed (using both SA and EA) on classical optimization problems like TSP and 0/1 Knapsack and other real-life problems in the domain of CAD for VLSI optimizations, have demonstrated the efficacy of the proposed frameworks.

References

1. Boddy, M., Dean, T.: Deliberation scheduling for problem solving in time-constrained environments. Artificial Intelligence 67, 245–285 (1994)
2. Aine, S.: New Approaches to Design and Control of Anytime Algorithmd. PhD thesis, Indian Institute of Technology Kharagpur, Department of Computer Science and Engineering, IIT Kharagpur 721302 India (2008)
3. van Laarhoven, P., Aarts, E.: Simulated Annealing: Theory and Applications. Kluwer, Dordrecht (1992)
4. Goldberg, D.E.: Genetic Algorithms in Search, Optimization and Machine Learning. Addison-Wesley Longman Publishing Co., Inc., Boston (1989)
5. Pohl, I.: Heuristic search viewed as path finding in a graph. Artif. Intell. 1(3), 193–204 (1970)
6. Likhachev, M., Gordon, G.J., Thrun, S.: Ara*: Anytime A* with provable bounds on sub-optimality. In: Advances in Neural Information Processing Systems, vol. 16, MIT Press, Cambridge (2004)
7. Zhang, W.: Complete anytime beam search. In: Proceedings of 14th National Conference of Artificial Intelligence AAAI 1998, pp. 425–430. AAAI Press, Menlo Park (1998)
8. Zhou, R., Hansen, E.A.: Beam-stack search: Integrating backtracking with beam search. In: Proceedings of the 15th International Conference on Automated Planning and Scheduling (ICAPS 2005), Monterey, CA, pp. 90–98 (2005)
9. Pearl, J.: Heuristics: intelligent search strategies for computer problem solving. Addison-Wesley Longman Publishing Co., Inc., Boston (1984)
10. Dean, T., Boddy, M.: An analysis of time-dependent planning. In: Proceedings of 6th National Conference on Artificial Intelligence (AAAI 1988), St. Paul, MN, pp. 49–54. AAAI Press, Menlo Park (1988)
11. Hansen, E.A., Zilberstein, S.: Monitoring and control of anytime algorithms: A dynamic programming approach. Artificial Intelligence 126(1-2), 139–157 (2001)