A Survey of Heuristics for Domain-Independent Planning

Ruishi Liang
School of Computer Engineering, Zhongshan Institute
University of Electronic Science and Technology of China, Zhongshan 528402, China
Email:liangruishi@gmail.com

***Abstract*—Increasing interest has been devoted to Planning as Heuristic Search over the years. Intense research has focused on deriving accurate heuristics in polynomial computational time for domain-independent planning. This paper reports on an extensive survey and analysis of research work related to heuristic derivation techniques for state space search planning, as well as other planning paradigms. Survey results reveal that heuristic techniques have been extensively applied in many efficient planners and result in impressive performances. We extend the survey analysis to suggest promising avenues for future research in heuristic derivation and heuristic search techniques.**

***Index Terms*—AI planning; domain-independent planning; heuristic search; heuristic derivation;**

I. Introduction

AI Planning is known to be PSPACE-complete [1] even in the simple STRIPS set [2]. Over the last few years a significant increase of the efficiency of planning system has been achieved with the evolution of biennial International Planning Competitions (IPC) [3].

One currently very successful trend in deterministic fully-automated planning is heuristic state space search. Specifically, Heuristic search-based planners perform a heuristic forward or backward search in the space of world states to find a path from the start state to a goal state. HSP is known as the first heuristic search-based planner which was introduced by Bonet et al. and competed in AIPS-1998 [4][5]. The success of HSP has inspired the development of many efficient planners, including FF [6][7], LPG[8][9], Fast Downward [10] and LAMA [11][12]. Several of them entered the International Planning Competitions. FF was in particular awarded for outstanding performance at IPC-2 and was the top performance planner in the STRIPS track of IPC-3. Following in the footstep of planner FF, LPG, Fast Downward and LAMA won the classical track of IPC-3, IPC-4 and IPC-6, respectively.

The techniques of heuristic search are also applied in other planning paradigms, in addition to deterministic state space planning. For example, numerical planning (Metric-FF[13], LPG-TD[9]), uncertainty planning (MBP[14]), compilation based planning (MIPS[15]). The performances of such planning paradigms benefit a lot from heuristic search.

Existing heuristics fell into two categories: on one hand, one can focus on deriving an inadmissible heuristic for satisficing planning by some fast approaches, e.g., [6][10][11]. On the other hand, on can perform a complete and accurate analysis to derive admissible heuristic for optimal planning, e.g., [16][17].

TABLE I.
CLASSIFICATION OF HEURISTICS

Heuristic admissibility	Planning paradigms	
	State-space planning	Other planning paradigms
Inadmissible	Additive heuristic	LPG, MFF
	Relaxed planning graph heuristic	
	Landmark heuristic	
	Casual graph heuristic	
Admissible	Critical path heuristic	BDD
	Pattern database heuristic	FD
	Abstraction heuristic	
	Landmark heuristic	

As a primary conclusion, we suggest a rough classification of heuristics (Table I). A heuristic is classified according to its admissibility as well as its application in planning paradigms. The above listed heuristics obviously represent existing best heuristics.

This paper studies heuristic techniques used in various planning paradigms. First, several classes of heuristics applied in state space planning will be analyzed, they are as follows: delete-relaxation heuristic, relaxed planning graph heuristic, critical path heuristic, causal graph heuristic, abstraction heuristic and landmark heuristic. Then the heuristic techniques used in non-state space planning paradigms will be discussed.

Manuscript received October 15, 2011; revised October 30, 2011; accepted October 30, 2011.
Corresponding author: liangruishi@gmail.com.

II. Heuristics for State-Space Planning

To introduce the background and notations of heuristic and planning model, we first identify the concept of heuristic and then briefly review STRIPS planning task definitions. Several classes of heuristics will be illustrated in this section later.

A. Heuristic

The notion of *heuristic* are strategies using readily accessible, though loosely applicable, information to control problem solving in human beings and machines [18]. Theoretically, there are two classes of problems where heuristic technique will be taken into account: first, a problem may have not a precise answer, then heuristic is used to find a most possible solution; Second, a problem has a precise answer but exhaustive search or blind search is impractical to solve it. Herein, heuristic method plays an important role in such situations by bringing experience-based techniques into solving problems.

A *heuristic* or *heuristic function* is said to be *admissible* if it is no more than the lowest-cost path to the goal, i.e., it never overestimates the cost of reaching the goal [19]. Specifically, let h be a heuristic function, for any state s, if $h(s) \leq distance\text{-}to\text{-}goal(s)$, then h is admissible. If a heuristic is not admissible, then it is *inadmissible*.

The common key idea to derive any heuristic in AI research is *relaxation*, i.e., ignoring some constraints in original complex problem *CP* to obtain a simpler problem *SP* which is easy to solve in polynomial time complexity. See figure 1.

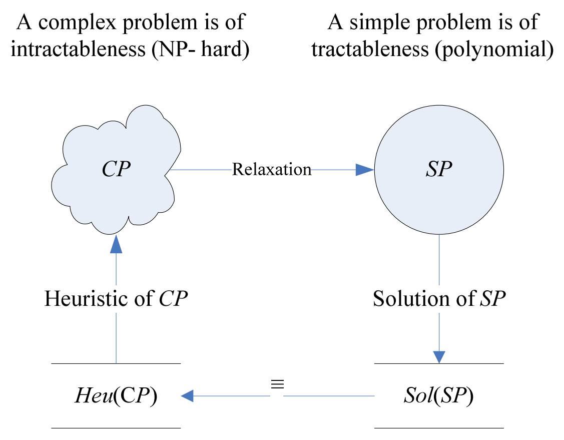

A complex problem is of intractableness (NP- hard)
A simple problem is of tractableness (polynomial)
CP
Relaxation
SP
Heuristic of CP
Solution of SP
Heu(CP)
≡
Sol(SP)

Figure 1. The relaxation idea to derive heuristic in AI research

B. STRIPS Planning Task

In propositional STRIPS planning, all construct are based on logic propositions. Given a set of propositions F, a world state s is a set of (true) propositions. An action a is given as a triple of proposition sets, $a = (pre(a), add(a), del(a))$, where $pre(a) \subseteq F$ is the preconditions of a, and $add(a) \subseteq F$ and $del(a) \subseteq F$ are the add list and the delete list, respectively. Given a world state s and an action a, the result of executing a in s, namely $result(s, a)$, is $result(s, a) := s \cup add(a) \setminus del(a)$, if the action is applicable in s, i.e., $pre(a) \subseteq s$. Otherwise, $result(s, a)$ is undefined. The set of all applicable actions in state s is denoted as $App(s)$. The result of executing an action sequence $<a_1, a_2,\ldots, a_N>$ in a state s is recursively defined by $result(s, <a_1, a_2,\ldots, a_N>) := result(result(s, a_1), <a_1, a_2,\ldots, a_N>)$, and $result(s, <>) := s$.

Definition 1. A STRIPS planning task $T = (F, A, I, G)$ is a tuple where F is the set of logic propositions, and A is the set of grounded actions, and I and G are initial state and goal state, respectively.

An action sequence $<a_1, a_2,\ldots, a_N>$ is a plan for T if and only if $G \subseteq result(I, <a_1, a_2,\ldots, a_N>)$.

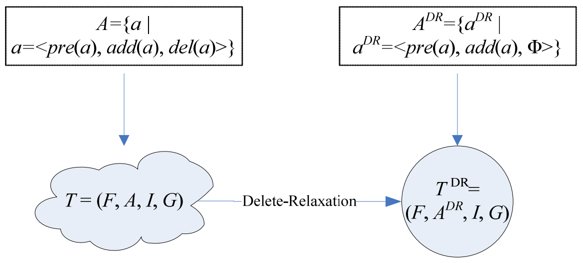

A={a | a=<pre(a), add(a), del(a)>}
A^DR={a^DR | a^DR=<pre(a), add(a), Φ>}
T = (F, A, I, G)
Delete-Relaxation
T^DR= (F, A^DR, I, G)

Figure 2. The core of Delete-Relaxation heuristic: neglect delete lists

C. Delete-Relaxation Heuristic

In classic planning, since $result(s, a) := s \cup add(a) \setminus del(a)$, a very intuitive relaxation idea for the classic planning search is to neglect $del(a)$. This simplified $result(s, a)$ involves only a monotonic increase in the number of propositions from s to $result(s, a)$. Hence, it is easier to compute distances to goals with such a simplified *result* mapping. See figure 2.

In the following, two well-known delete-relaxation heuristics which were first developed in HSP and HSPr by Halsum et al. will be analyzed: one is additive heuristic h^{add}, and the other is maximum heuristic h^{max} [4][5].

Given a STRIPS task $T = (F, A, I, G)$. Let s be a search state in T and p be an atom, the distance from s to p, denoted by $d^a(s, p)$, is the minimum number of actions required to reach from s to a state containing p.

$$d^{a}(s,p)=\begin{cases}0, if \ \ p\in s,\\ \infty, if \ \ \forall a\in A, p\notin add(a), and \ \ p\notin s,\\ \min\{1+d^{a}(s,prec(a)) \mid p\in add(a), a\in A\}, otherwise.\end{cases}$$

The distance from s to a set of propositions F is computed as follows:

$$d^a(s,F) = \begin{cases} 0, if\ F \subseteq s, \\ \sum_{p \in F} d^a(s,p), otherwise. \end{cases}$$

As a result, additive heuristic is defined as follows:

$$h^{add}(s) \equiv d^a(s,G)$$

The additive heuristic is not admissible, which motivates the idea of maximum heuristic. Similar to additive heuristic, the distance from a state s to an atom p, denoted by $d^x(s, p)$, is the minimum number of actions required to reach from s to a state containing p.

$$d^x(s,p) = \begin{cases} 0, if\ p \in s, \\ \infty, if\ \forall a \in A, p \notin add(a), and\ p \notin s, \\ \min\{1 + d^x(s, prec(a)) \mid p \in add(a), a \in A\}, otherwise. \end{cases}$$

However, the distance from s to a set of propositions F is different from the equation in additive heuristic framework, it is computed as follows:

$$d^x(s,F) = \begin{cases} 0, if\ F \subseteq s, \\ \max\{d^x(s,p) \mid p \in F\}, otherwise. \end{cases}$$

As a result, the maximum heuristic h^{max} is defined as follows:

$$h^{\max}(s) \equiv d^x(s,G)$$

As a generalization, the additive heuristic combines the costs to all subgoals as the distance estimator of a search state to goal state, while the maximum heuristic considers only the relaxed distance to the most difficult subgoal as heuristic of a search state.

Based on additive heuristic and maximum heuristic, HSP performs a progression search from initial state to goal state by standard hill-climbing search algorithm, while HSP2 does similar forward heuristic search but with weight A* algorithm. In contrast, HSPr is a regression planner which performs the search backward from the goal state rather than forward from initial state in order to avoid re-computation of the estimate costs for all atoms in every encountered new state. Overall, the HSP planner family got an impression result in AIPS 1998 planning contest and inspired the developments of various distinguished heuristic techniques for last decade.

Liu et al. studied a novel approach named PINCH to speed up the calculation of delete-relaxation heuristics [20]. The core operations are ordering the value updates and reusing information from the calculation of previous heuristic values. The experimental result showed that PINCH method saved at most 80% heuristic computation time compared to HSP and HSP 2.

D. Relaxed Planning Graph Heuristic

Relaxed Planning Graph (RPG) heuristic was introduced firstly in Fast-Forward (FF) planner [6][7], where FF is a progression heuristic planner, searching the space of world states of a planning task in the forward direction, guided by Relaxed Planning Graph heuristic functions. RPG heuristic is also based on the delete-relaxation idea, however, different from additive heuristic and maximum heuristic that defined in recursive formulation, it estimates the goal distance of search state through defining a relaxation of original planning task by ignoring the delete effects of the actions in A and solving the relaxed task explicitly.

Specifically, a planning graph for the relaxed planning task will be constructed by the GraphPlan [21] algorithm first. Then a non-optimal relaxed plan, a solution to the relaxed task, can be extracted from the planning graph in polynomial time, which is the basis of heuristic function.

Definition 2. Given a STRIPS task $T = (F, A, I, G)$. The relaxation a^+ of an action $a \in A$, $a = (pre(a), add(a), del(a))$, is defined as $a^+ = (pre(a), add(a), \varnothing)$. Accordingly, the relaxation T^+ of T is (F, A^+, I, G), where $A^+ = \{ a^+ \mid a \in A \}$.

An action sequence $<a_1, a_2,\dots, a_N>$ is a relaxed plan for T if $<a_1^+, a_2^+,\dots, a_N^+>$ is a plan for T^+.

Definition 3. Given a STRIPS task $T = (F, A, I, G)$. Let s be a search state in T. The relaxed planning graph (RPG) for the relaxed task (F, A^+, I, G) is a layered graph alternating between proposition levels and action levels, which is represented as a sequence $P_0, A_0,\dots, P_i, A_i,\dots, A_{k-1}, P_k$.

$$P_i := \begin{cases} s & , if\ i = 0 \\ P_{i-1} \bigcup \{add(a) \mid pre(a) \subseteq P_{i-1}\}, & if\ 1 \le i \le k \end{cases}$$

$$A_i := \{a \in A^+ \mid pre(a) \subseteq P_i\}, 0 \le i < k.$$

where k is the final layer of the relaxed planning graph.

After constructing the RPG for a search state s, FF starts at the final layer of RPG and uses a backtrack-free procedure that extracts a sequence of actions that correspond to a successful relaxed plan for s. During the extraction process, two kinds of sets are maintained: the sequence of sub-goal sets $G_1,\dots, G_k$ that represent the sub-goals first appearing in the respective levels $P_1,\dots, P_k$; the sequence of solution action set $O_0,\dots, O_{k-1}$ that represent the actions achieving the sub-goals in $G_1,\dots, G_k$. Actually, the sequence $< O_0,\dots, O_{k-1}>$ is the relaxed plan for s, and the total number of actions in the sequence is the heuristic value for s, i.e.,

$$h(s) = \Sigma_{i=0}^{t-1} \mid O_i \mid .$$

If no relaxed plan can be extracted, the heuristic value of s is set to ∞.

Based on RPG and relaxed plan, a pruning strategy called helpful actions is proposed in FF. Formally, the set of helpful actions to s is defined as

$$HPA(s) := \{a \in A \mid pre(a) \subseteq s \wedge add(a) \cap G_1 \neq \varnothing\}.$$

where G_1 is the set of sub-goals constructed at layer P_1 of RPG.

The core search procedure used in FF is a variation of classic hill climbing local search algorithm, namely, enforced hill-climbing search algorithm. It performs a manner of systematic search to find better states but in local search state spaces.

If the local search in FF fails to find better states, i.e., it encounters a local minimal, then a complete heuristic search algorithm is activated to solve the planning task from scratch.

The heuristic search framework combining with relaxed planning graph heuristic used in FF is presented in Figure 3.

```
Input: a planning task <A, I, G>
Output: a solution plan π or Failure
  π := < >;
  s := I;
  h(s) is heuristic estimator for state S computed by
  relaxed planning graph heuristic procedure
  while h(s) != 0 do
      perform Enforced Hill-Climbing search to find
      a better state s' with h(s') < h(s)
      if no such state s' can be found then
          π = active Greedy Best-First search from
          scratch to sovle the planning task;
          return π ;
    endif
    add the actions on the path to s' at the end of π
    s = s';
  endwhile
  return π ;
```

Figure 3. The heuristic search framework combining with relaxed planning graph heuristic used in FF

The key of relaxed planning graph heuristic is the construction and extraction of a relaxed plan corresponding to a relaxed planning task, i.e., no delete effects are taken into account during computations. However, as pointed out in the literature [22], in a number of benchmark domains, e.g., Blocksworld, Gridworld, such relaxed plan provides poor guidance to heuristic search to reach the goal. Hence, Yoon et al. [22] proposed a leaning approach to improve relaxed planning graph heuristic by partially incorporate delete list information. Specifically, the approach leverage the delete lists for the actions in a relaxed plan by an inductive way, i.e., learning a linear regression function that approximates the differences between relaxed plan length and the observed distance-to-goal of states in the available training plans. The value of regression function is seen as a kind of revision to basic relaxed plan length. As a result, the improved heuristic is the sum of basic relaxed plan length and regression function.

The idea of relaxed planning graph heuristic is also applied widely in other planning models besides in STRIPS set, such as numerical planning, contingent planning, conformant planning, etc. See section III in detail.

E. Critical Path Heurisitc

The notation of *critical path heuristic* was refined by Helmert and Domshlak [23]. Indeed, it represents the h^m heuristic family which was first formulated by Haslum and Geffner [24].

As mentioned in delete-relaxation heuristic, the max heuristic is admissible but less informative than the additive heuristic because it only considers one most difficult atom when computing heuristic for every state. A nature extension to max heuristic is to take into account more atoms in each iteration, which results in the generalization of h^m heuristic family. Technically, the heuristic value of an intermediate search state to goal state is assigned with the highest cost of reaching a subset with at most m satisfied atoms within the search state. The formal definition of h^m heuristic family is as follows.

The distance from a state s to a set of atoms F, denoted by $d(s, F)$, is the minimum number of actions required to reach from s to a state containing F.

$$d(s,F)=\begin{cases}0, \textit{if } F\subseteq s,\\ \infty, \textit{if } \forall a\in A, a \textit{ is not relevant } F,\\ \min\{1+d(s,\gamma^{-1}(F,a)) \mid a \textit{ is relevant } F\}, \textit{otherwise.}\end{cases}$$

Then the estimated distance from s to goal state G is computed as follows:

$$d^m(s,G)=\begin{cases}0, \textit{if } G\subseteq s,\\ \infty, \textit{if } \forall a\in A, a \textit{ is not relevant } G,\\ \min_a\{1+d(s,\gamma^{-1}(G,a)) \mid a \textit{ is relevant } G \textit{ and } |G|\le m\},\\ \max_F\{d^m(s,F) \mid F\subseteq G \textit{ and } |F|=m\}, \textit{otherwise.}\end{cases}$$

Given the estimated distance, the h^m heuristic of a search state s is defined as follows:

$$h^m(s)\equiv d^m(s,G)$$

The computation complexity is polynomial time in the number of m from a theoretic point of view.

Although the extension seems to be slight and the theoretic computation is in polynomial time complexity, it requires a feasible computation method for h^m heuristic family. The main contribution of h^m heuristic family proposed by Haslum and Geffner is mapping shortest-path problem in state space into suitably defined shorted-path problems in atom space. Another interested contribution of h^m heuristic family lies in its abilities to deal with planning with action cost as well as parallel planning.

h^m heuristic family provides more informative estimate for a search state than additive heuristic and max heuristic. Generally, the bigger value the parameter m takes, the better heuristic we can obtain. A main disadvantage for large m value is the time-consuming computation process for each search state because more subgoal interactions are taken into account. Hence, as a trade-off between the accuracy of the heuristic and its computational cost, it is reasonable to set m less than or equal with two.

Haslum et al. also developed a novel admissible heuristic based on h^m heuristic, named *additive h^m* heuristic [25]. The main improvement for h^m heuristic is to partition the set of actions A into n disjoint set $A_1,\ldots,A_n$, and then each individual heuristic $h^m{}_{Ai}$ will be compute separately which requires much less time than the computation of original h^m heuristic constructed on the set of actions A. Finally, additive h^m heuristic is assigned with the sum of all $h^m{}_{Ai}$ value.

F. Causal graph heuristic

Causal graph heuristic was proposed by Helmert in planner Fast Downward [10] that was developed based on SAS+ planning formalism where state is represented as multi-valued variable. Generally speaking, causal graph is a directed graph constructed based on domain transition graphs and causal dependencies imposed on domain transition graphs. The planning process is then viewed as search in causal graph of a planning task. Let *s* be a search state, the causal graph heuristic *h* is computed as follows:

$$h\,(s) = \sum_{v \in V} cost_v\,(s(v), g(v))$$

where *v* is a state variable and $cost_v\,(s(v), g(v))$ is the cost estimate from $s(v)$ to $g(v)$, i.e. the cost of changing the value of *v* from $s(v)$ to $g(v)$. The planner uses causal graph heuristic to guide heuristic search in causal graph.

G. Abstraction Heuristic

Besides delete-relaxation idea in heuristic research for classic planning, another promising heuristic method is *abstraction* [16][26]. From conceptual view, abstraction is a way to make the search space of abstracted problem smaller than that of original problem. Since the search space is small enough, it is feasible to derive heuristic through computing the abstracted problem by various approaches or algorithms. See figure 4. This is the common theoretic basis similar with relaxation idea to derive heuristic in AI research. However, the main difference from relaxation idea is that an abstracted problem is not required to satisfy polynomial property, i.e., polynomial computation cost. Often, an abstracted problem will be pre-computed before search by blind search algorithm in order to obtain an optimal solution which is the tighter lower bound heuristic on the real optimal solution of original problem. Herein, abstraction heuristic is admissible.

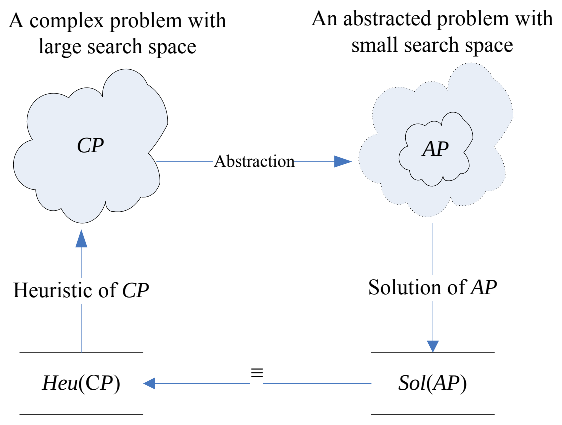

A complex problem with large search space
An abstracted problem with small search space
CP
Abstraction
AP
Heuristic of CP
Solution of AP
Heu(CP)
≡
Sol(AP)

Figure 4. The abstraction idea to derive heuristic in AI research

Generally speaking, an abstraction of a search state is given by a mapping χ: $S \rightarrow S^{\chi}$, where S is the set of the states in original planning task and S^{χ} is the set of the abstracted states in abstract planning task. The definition of abstraction thus can be generalized to planning task, or states transition system in more general form. The abstraction heuristic value $h^{\chi}\,(s)$ is then the cost from $\chi\,(s)$ to the closest goal state of the abstract planning task.

The first attempt to bring abstraction heuristic into planning communication is the derivation of pattern database heuristic (PDB) by Edelkamp in the literature [16] , which are based on projecting the planning task onto a subset of its state variables and then explicitly searching for optimal plans in the abstract space. Pattern database heuristics [27] [28][29] are well known for its distinguished performance in solving Rubik's Cube problems, 15-Puzzle and other similar combinatorial problems. By properly constructing state space and planning space, it is possible to derive PDB heuristic in the context of deterministic planning.

Following PDB heuristic, several other abstraction heuristics have been proposed, for example, constrained PDB heuristic [30], symbolic pattern database heuristic [31], merge-and-shrink abstraction heuristic [26], explicit-state abstraction heuristic [32], implicit abstraction heuristic [33], and structural patterns [34]. Symbolic PDB heuristic uses Boolean function to represent heuristic function in the context of symbolic planning space. Exactly speaking, symbolic PDB is constructed in the form of binary decision diagram (BDD). Based on symbolic representation of heuristic function and planning space, explicit pattern database search algorithm and symbolic pattern database search algorithm thus can be both incorporated into the symbolic planning framework. Merge-and-shrink abstraction derives heuristic based on the SAS+ formulation consisting of two parts: the merging strategy that decides to choose which two abstractions to compute their synchronized product, and the shrinking strategy that chooses which abstraction to compute its homomorphism. Over the years, abstract heuristics have shown to be very effective in several hard search problems, including cost-optimal planning [35].

H. Landmark Heuristic

The notion of *landmarks* for deterministic planning are facts that must take place at some point in every solution plan for a given planning task [36]. For instance, on(A, B) is a goal in a Blocksworld task that has block A stacked on block B. It is evident that clear(B) must hold at some point for the goal on(A, B) to be achieved, and thus clear(B) is a landmark for that task. Goals are trivially landmarks, and thus on(A, B) is a landmark as well. There is an ordering between these two landmarks. Due to different constraints imposed on orderings, various ordering relationships will be defined precisely [36][37]. Given an instantiated planning task, all landmarks and orderings can be extracted theoretically, though it is indeed PSPACE-hard. A number of approximate methods have been proposed to obtain landmarks and orderings as much as possible in preprocess before planning.

The *Landmark heuristic* was first introduced by Richter et al. [12]. Since landmark is a intermediate (or goal-level) fact that must be achieved in order to reach a goal, a naive idea using landmark and ordering information to derive heuristic for a state *s* is to compute the landmarks that still need to be achieved from *s* onwards. Given a planning task *T*, the set of extracted landmarks *L* and the set of orderings between landmarks *D* are pre-computed before planning. Let *s* be a search state and π be a path, i.e., a sequence of states, then the formal definition of landmark heuristic *h* is defined as follows:

$h(s) = (L - Accepted(s, \pi)) \cup ReqAgain(s, \pi)$

where $Accepted(s, \pi) \subseteq L$ and $ReqAgain(s, \pi) \subseteq Accepted(s, \pi)$ are the sets of accepted and required again landmarks, respectively. A landmark is accepted if it is true at some time along π. An accepted landmark is required again if (i) it does not hold in , and (ii) it is ordered greedy necessarily before some landmark which is not accepted. The number *L* (*s*, π) is then the heuristic value assigned to state *s*.

The LAMA planner [11] developed by Richter and Westphal employed such landmark based heuristic in its search framework and was the clear winner of the Sequential Satisficing Track at the 2008 International Planning Competition. in addition to derive heuristic, landmark information is also used to compute preferred operators. Karpas and Domshlak [38] also proposed a methodology for deriving admissible heuristic estimates for cost-optimal planning from a set of planning landmarks. The resulting heuristics fall into a novel class of multi-path dependent heuristics.

I. Summary

So far, we have discussed several heuristic derivation techniques separately. There are some incomplete comparisons on accuracy of these heuristics [23][39]. A new trend is to combine different heuristics to form a more informative heuristic estimate. For example, Roger et al. described several methods for combining heuristics estimates in satisficing planning framework [40]. Helmert et al. obtained landmark cut heuristic by analyzing the connections between four classes of heuristics [23]. Domshlak et al. proposed a way to bring landmarks information into abstraction [41].

III. Heuristic for other planning paradigms

Besides in state space planning, heuristic techniques are also applied in other planning paradigms. Metric-FF [13] can be seen as a numeric version of FF, thus it is also a heuristic planner. The heuristic of Metric-FF significantly extends the ability of relaxed planning graph heuristic to deal with numeric information. The planner LPG-TD derives heuristic in action graph which is able to handle numeric information [9]. MBP is a non-deterministic planner [14] performs heuristic search in and/or graph. The compilation based planner MIPS [15] uses heuristic techniques in BDD search space in order to compact large search space. Other planners employing heuristic techniques include SATPLAN [42], SAPA^{PS}, Yochan^{PS} [43] and so on. The performances of those planning paradigms benefit a lot from heuristic search.

VI. Conslusion

As a fundamental problem-solving tool in artificial intelligence research as well as in computer science, heuristic techniques are widely used to solve lots of complicated problems. Researchers pay much effort to derive accurate heuristics, to construct compact search spaces, and to develop effective guided search algorithms. So far, a number of generic heuristic search algorithms are developed by researchers which fell into two categories: complete search and local search. Examples of complete search algorithms used in planning include A*, WA*, greedy best-first search, Anytime Heuristic [43], BDDA* [44], Examples of local search algorithms include hill climbing, Enforced Hill Climbing, stochastic local search, and so on.

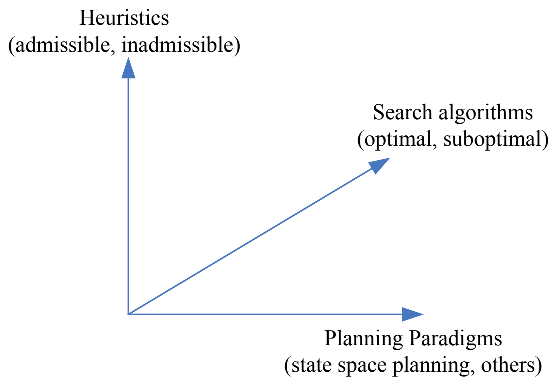


Figure 5. The abstraction idea to derive heuristic in AI research

Planning as heuristic search has been a dominant approach in planning community in last decade. A number of heuristic based planners are developed and exhibit distinguished planning performances, such as HSP, FF, Fast Downward, LAMA, and so on. The hot topics of heuristic planning are as follows: heuristic derivation techniques to obtain more accurate heuristic evaluation with less computation cost, search algorithms to guide search process more quickly, and planning paradigms to better match with problem structures and/or solution requirements. That is to say, due to problem structures and/or different solution requirements, admissible or inadmissible heuristics, optimal solution or suboptimal solution, and state space planning or other planning paradigms will be considered specifically.

We have reported on an extensive survey and analysis of research work related to heuristic derivation techniques for state space search planning and other planning paradigms. Survey results reveal that heuristic techniques have been extensively applied in many efficient planners and result in impressive performances. Further research includes deriving more accurate heuristics through exploiting structure information in greater depth, and combining different heuristic estimates into a consistent framework to improve the informativeness of the heuristics.

ACKNOWLEDGMENT

The authors wish to thank the anonymous reviewers for their helpful suggestions to improve the manuscript. This work was supported in part by the Scientific Research Foundation of Zhongshan Institute, University of Electronic Science and Technology of China under Grant No. 410YKQ03.

REFERENCES

[1] Helmert M. 2003. Complexity results for standard benchmark domains in planning. Artificial Intelligence 143(2):219~262.

[2] Fikes, R. Nilsson, N. 1971. STRIPS: A new approach to the application of theorem proving to problem solving. Artificial Intelligence 2: 189-208.

[3] International Planning Competition Archives official website: http://ipc.icaps-conference.org/.

[4] Bonet, B., Geffner, H. Planning as Heuristic Search. 2001. Artificial Intelligence 129:5-33.

[5] Bonet, B., Geffner, H. Heuristic Search Planner 2.0. AI Magazine 2001, 22(3): 77-80.

[6] Hoffmann, J. and Nebel, B. 2001. The FF planning system: Fast plan generation through heuristic search. Journal of Artificial Intelligence Research 14: 253-302.

[7] Hoffmann, J. Where "Ignoring Delete Lists" Works: Local Search Topology in Planning Benchmarks. Journal of Artificial Intelligence Research, 2005(24): 685-758.

[8] Gerevini, A., Serina, I. LPG: a Planner based on Local Search for Planning Graphs. Proceedings of the 6th International Conference on Artificial Intelligence Planning and Scheduling (AIPS'02), 2002: 12-22.

[9] Gerevini, A., Saetti, A., Serina, I. An Approach to Efficient Planning with Numerical Fluents and Multi-Criteria Plan Quality. Artificial Intelligence, 2008, 172(8-9):899-944.

[10] Helmert, M. 2006. The fast downward planning system. Journal of Artificial Intelligence Research, 26: 191--246.

[11] Richter, S. and Westphal, M. The LAMA planner: Guiding cost-based anytime planning with landmarks. Journal of Artificial Intelligence Research, 2010, 39: 127-177.

[12] Richter, S., Helmert, M. and Westphal, M. 2008. Landmark revisited. In Proc. AAAI 2008, 975–982.

[13] Hoffmann, J. The Metric-FF Planning System: Translating "Ignoring Delete Lists" to Numeric State Variables. Journal of Artificial Intelligence Research, 2003,20: 291-341.

[14] Bertoli, P. and Cimatti, A. and Pistore, M. and Roveri, M. and Traverso, P. 2001. MBP: a model based planner. In Proc. of the IJCAI'01 Workshop on Planning under Uncertainty and Incomplete Information.

[15] Edelkamp, S. and Helmert, M. 2001. The model checking integrated planning system (MIPS). AI Magazine, 22(3):67-71.

[16] Edelkamp, S. 2001. Planning with pattern databases. In Proc. ECP 2001, 13–24.

[17] Karpas, E., Domshlak, C. 2009. Cost-optimal planning with landmarks. In Proceedings of the 21st International Joint Conference on Artificial Intelligence (IJCAI 2009), pp. 1728–1733.

[18] Pearl, J. 1983. Heuristics: intelligent search strategies for computer problem solving. New York, Addison-Wesley, 1983.

[19] Russell, J., Norvig, P. 2002. Artificial Intelligence: A Modern Approach. Prentice Hall. 2002.

[20] Liu, Y. and Koenig, S. and Furcy, D. 2002. Speeding up the calculation of heuristics for heuristic search-based planning. In Proceedings of the National Conference on Artificial Intelligence (AAAI 2002), 484-491.

[21] Blum A., Furst M. Fast Planning Through Planning Graph Analysis. Artificial Intelligence, 1997, 90:281--300.

[22] Yoon, S. and Fern, A. and Givan, R. 2006. Learning Heuristic Functions from Relaxed Plans. In Proc. of International conference on automated planning and scheduling (ICAPS 2006), 162-171.

[23] Helmert, M. and Domshlak, C. 2009. Landmarks, Critical Paths and Abstractions: What's the Difference Anyway. In Proc. of International conference on automated planning and scheduling (ICAPS 2009), 162-169.

[24] Haslum, P. and Geffner, H. 2000. Admissible heuristics for optimal planning. In Proc. of AIPS 2000, 140-149.

[25] Haslum, P. and Bonet, B. and Geffner, H. 2005. New admissible heuristics for domain-independent planning. In Proceedings of the National Conference on Artificial Intelligence (AAAI 2005), 1163-1168.

[26] Helmert, M.; Haslum, P.; and Hoffmann, J. 2007. Flexible abstraction heuristics for optimal sequential planning. In Proc. ICAPS 2007, 176–183.

[27] Korf, R.E. and Taylor, L.A. 1996. Finding optimal solutions to the twenty-four puzzle. In Proceedings of the National Conference on Artificial Intelligence (AAAI 1996), 1202-1207.

[28] Korf, R.E. 1998. Finding optimal solutions to Rubik's Cube using pattern databases. In Proceedings of the National Conference on Artificial Intelligence (AAAI 1998), 700-705.

[29] Korf, R.E. and Felner, A. 2002. Disjoint pattern database heuristics. Artificial Intelligence, 2002, 134(1-2): 9-22.

[30] Haslum, P. and Botea, A. and Helmert, M. and Bonet, B. and Koenig, S. 2007. Domain-independent construction of pattern database heuristics for cost-optimal planning. In Proceedings of 22rd AAAI Conference on Artificial Intelligence (AAAI'07), 1007- 1012.

[31] Edelkamp, S. 2002. Symbolic pattern databases in heuristic search planning. In Proc. Artificial Intelligence Planning and Scheduling (AIPS 2002), 274-283.

[32] Helmert, M. and Haslum, P. and Hoffmann, J. 2008. Explicit-state abstraction: A new method for generating heuristic functions. In Proceedings of 23rd AAAI Conference on Artificial Intelligence (AAAI'08), 1547-1550.

[33] Katz, M., and Domshlak, C. Implicit Abstraction Heuristics. Journal of Artificial Intelligence Research, 2010, 39: 51-126.

[34] Katz, M., and Domshlak, C. 2009. Structural-pattern databases. In Proc ICAPS 2009, 186–193.

[35] Katz, M., Domshlak, C. (2010). Optimal admissible composition of abstraction heuristics. Artificial Intelligence, 174, 767–798.

[36] Hoffmann J and Porteous J and Sebastia L. Ordered landmarks in planning. Journal of Artificial Intelligence Research, 2004, 22: 215-278.

[37] Koehler, J. and Hoffmann, J. 2000. On reasonable and forced goal orderings and their use in an agenda-driven planning algorithm. Journal of Artificial Intelligence Research, 2000, 12: 338-386.

[38] Karpas, E., Domshlak, C. (2009). Cost-optimal planning with landmarks. In Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI-09), pp. 1728–1733, Pasadena, CA, USA.

[39] Helmert, M. and MattmUller. 2008. Accuracy of admissible heuristic functions in selected planning domains. In Proceedings of 23rd AAAI Conference on Artificial Intelligence (AAAI'08), 938-943.

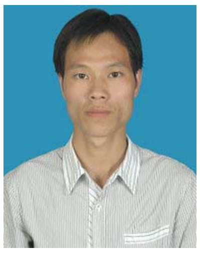
[40] Roger, G. and Helmert, M. 2009. Combining heuristic estimators for satisficing planning. In ICAPS 2009 Workshop on Heuristics for Domain-Independent Planning, 43-48.

[41] Domshlak, C., Katz, M., Lefler, S. 2010. When abstractions met landmarks. In Proc ICAPS 2010, pp. 50–56.

[42] Kautz, H. and Selman, B. 2004. SATPLAN04: Planning as satisfiability. In Proc. ICAPS 2004.

[43] Benton, J. and Do, M. and Kambhampati, S. 2009. Anytime heuristic search for partial satisfaction planning. Artificial Intelligence, 173(5-6): 562-592.

[44] Jensen, R.M. and Veloso, M.M. and Bryant, R.E. 2008. State-set branching: Leveraging BDDs for heuristic search. Artificial Intelligence, 172(2-3): 103-139.



Ruishi Liang, is currently a lecturer of Computer Science at Zhongshan Institute, University of Electronic Science and Technology of China. His main research interests include AI planning, heuristic search in planning. He received his Ph.D degree from Sun Yat-sen University in mainland of China in 2010, and was a member of AI planning group directed by Professor Yunfei Jiang from 2005 to 2010. In last five years, he developed two deterministic planner HQFF and ASOP which both are competitive with some top performance planners in satisficing track at the International Planning Competitions (IPC), and contributed many publications on the area of AI planning, especially on the topics of heuristic search based planning and subgoal orderings for planning. Now he directs the Scientific Research Foundation of Zhongshan Institute, University of Electronic Science and Technology of China under Grant No. 410YKQ03, and is a member of National Natural Science Foundation of China under Grant No. 60970042. His email address is liangruishi@gmail.com.