Selecting Actions and Making Decisions: Lessons from AI Planning

Héctor Geffner
Departamento de Tecnología
ICREA – Universitat Pompeu Fabra
Barcelona 08003, SPAIN
hector.geffner@upf.edu

Abstract

Humans encounter a huge variety of problems which they must solve using general methods. Even simple problems, however, become computationally hard for general solvers if the structure of the problems is not recognized and exploited. Work in Artificial Intelligence Planning and Problem Solving has encountered a similar difficulty, leading in recent years to the development of well-founded and empirically tested techniques for recognizing and exploiting structure, focusing the search for solutions in certain cases, and bypassing the need to search in others. These techniques include the automatic derivation of heuristic functions, the use of limited but effective forms of inference, and the compilation of domains, all of which enable a general problem solver to 'adapt' automatically to the task at hand. In this paper, I present the ideas underlying these new techniques, and argue for their relevance to models of natural intelligent behavior as well. The paper is not a review of AI Planning – a diverse field with a long history – but a personal appraisal of some recent key developments and their potential bearing on accounts of action selection in humans and animals.

1 Introduction

In the late 50's, Newell and Simon introduced the first AI planner – the General Problem Solver or GPS – as a psychological theory [Newell and Simon, 1958; 1963]. Since then, Planning has remained a central area in AI while changing in significant ways: it has become more mathematical (a variety of planning problems has been clearly defined and studied) and more empirical (planners and benchmarks can be downloaded freely, and competitions are held every two years), and as a result, new ideas and techniques have been developed that enable the automatic solution of large and complex problems [Smith, 2003].

AI Planning studies languages, models, and algorithms for describing and solving problems that involve the selection of actions for achieving goals. In the simplest case, in *classical planning*, the actions are assumed deterministic, while in *contingent planning*, actions are non-deterministic and there is feedback. In all cases, the task of the planner is to compute a plan or solution; the *form* and *cost* of these solutions depending on the model; e.g., in classical planning, solutions are sequences of actions and cost is measured by the number of actions, while in planning with uncertainty and feedback, solutions map states into actions, and cost stands for expected or worst-possible cost.

Planning is a form of 'general problem solving' over a class of models, or more precisely, a *model-based* approach to intelligent behavior: given a problem in the form of a compact description of the actions, sensors (if any), and goals, a planner must compute a solution, and if required, a solution that minimizes costs. Some of the models used in planning, as for example Markov Decision Processes (MDPs), are not exclusive to AI Planning, and are used for example in Control Theory [Bertsekas, 1995], Reinforcement Learning [Sutton and Barto, 1998], and Behavioral Ecology [Houston and McNamara, 1988; Clark, 1991] among other fields. What is particular about AI planning are the *languages* for representing these models, the *techniques* for solving them, and the ways these techniques are *validated*. Techniques do matter quite a lot: even simple problems give rise to very large state spaces that cannot be solved by exhaustive methods. Consider the well known Rubik Cube puzzle: the number of possible configurations is in the order of trillions, yet methods are known for solving it, even optimally, from arbitrary configurations [Korf, 1998]. The key idea lies in the use of *admissible heuristic functions* that provide an optimistic approximation of the number of moves to solve the problem from arbitrary configurations. These functions enable the solution of large problems, even ensuring optimality, by focusing the search and avoiding most states in the problem. Interestingly, recent work in planning has shown that such functions can be derived *automatically* from the problem description [Bonet and Geffner, 2001], and can be used to drive the search in problems involving uncertainty and feedback as well [Bonet and Geffner, 2000]. Such functions can be understood as a specific and robust form of *means-ends analysis* [Newell and Simon, 1958; 1963] that produces goal-directed behavior in complex settings even in the presence of large state and action spaces.

In this paper, we review some of the key computational ideas that have emerged from recent work in planning and problem solving in AI, and argue that these ideas, although

not necessarely in their current form, are likely to be relevant for understanding natural intelligent behavior as well. Humans encounter indeed a huge variety of problems which they must solve using general methods. It cannot be otherwise, because there cannot be as many methods as problems. Yet, simple problems become computationally hard for a general solver if the structure of the problems is not recognized and exploited. This is well known in AI, where systems that do not exhibit this ablity tend to be shallow and brittle. In the last few years, however, work in Planning and Problem Solving has led to well-founded and empirically tested techniques for recognizing and exploiting structure, focusing the search for solutions, and in certain cases, bypassing the need to search altogether. These techniques include the automatic derivation of heuristic functions, the use of limited but effective forms of inference, and the compilation of domains, all of which enable a general problem solver to 'adapt' automatically to the task at hand. Interestingly, the need for focusing the search for solutions has been recognized in a number of recent works concerned with natural intelligent behavior, where it has been related to the role of emotions in the appraisal and solution of problems. We will say more about this as well.

Since Newell's and Simon's GPS, the area of AI planning has departed from the original motivation of understanding human cognition to become the mathematical and computational study of the problem of selecting actions for achieving goals. Yet after all these years, and given the progress achieved, it is time to reflect on what has been learned in the abstract setting, and use it for informing our theories in the natural setting. This exercise is possible and may be quite rewarding. It parallels the approach advocated by David Marr, and echoed more recently by [Glimcher, 2003] and others in the Brain Sciences; namely: characterize *what* needs to be computed, *how* it can be computed, and how these computations are *approximated* in real-brains. The findings that we summarize below, aim to provide a partial account of the first two tasks.

A few methodological comments before proceeding. First about *domain-generality* vs. *domain-specificity* in action selection. I have said that humans are capable of solving a wide range of problems using general methods. This, however, is controversial. Both evolutionary psychologists [Tooby and Cosmides, 1992] and cognitive scientists from the 'fast and frugal heuristics' school [Gigerenzer and Todd, 1999] place an emphasis on modularity and domain-specificity. Others, without necessarily denying the role of specialization, postulate the presence of general reasoning and problem solving mechanisms as well, at least in humans (see for example [Stanovich, 2004]). We are not going to address this controversy here, just emphasize that 'general' and 'adapted' are not necessarily opposite of each other. Indeed, the work in AI planning is domain-independent, yet the recent techniques illustrate how a general problem solver can 'adapt' to a specific problem by recognizing and exploiting structure, for example, in the form of heuristic functions. These heuristics are indeed in line with the 'fast and frugal heuristics', the difference being that they are general and can be extracted automatically from problem descriptions.

Another distinction that is relevant for fitting the work in AI Planning within the broader work on Intelligent Behavior is the one between *finding solutions* vs. *executing solutions*. For many models, such as those involving uncertainty and feedback, the solutions, from a mathematical point of view, are functions mapping states into actions (these functions are called *closed-loop policies*, and in the partially observable case map actually *belief states* into actions; see below). These functions can be represented in many ways; e.g. as condition, action rules, as value functions, etc. Indeed, in what is often called *behavior-based AI* [Brooks, 1997], these solutions are encoded by hand for controlling mobile robots. In nature, similar solutions are thought to be encoded in brains but not by hand but by evolution. Representing and executing solutions, however, while challenging, is different than coming up with the solutions in the first place which is what AI Planning is all about. Whether this is a requirement of intelligent behavior in animals is not clear although it seems to be a distinctive feature of intelligent behavior in humans. Interestingly, in many cases, the same models can be used for both *understanding* the solutions found in nature, and for *generating* those solutions [McFarland and Bosser, 1993]. The interest in the latter case, however, is not only with the models but also with the algorithms needed for solving those models effectively. We thus consider both models and algorithms.

2 Models

Most models considered in AI Planning can be understood in terms of *actions* that affect the *state* of a system, and can be given in terms of

1. a discrete and finite state space S,
2. an initial state $s_0 \in S$,
3. a non-empty set of terminal states $S_T \subseteq S$,
4. actions $A(s) \subseteq A$ applicable in each non-terminal state,
5. a function $F(a, s)$ mapping non-terminal states s and actions $a \in A(s)$ into *sets* of states
6. action costs $c(a, s)$ for non-terminal states s, and
7. terminal costs $c_T(s)$ for terminal states.

In deterministic planning, there is a single predictable next state and hence $|F(a, s)| = 1$, while in non-neterministic planning $|F(a, s)| \geq 1$. In addition, in probabilistic planning (MDPs), non-deterministic transitions are weighted with probabilities $P_a(s'|s)$ so that $\sum_{s' \in F(a,s)} P_a(s'|s) = 1$. In general, action costs $c(a, s)$ are assumed to be positive, and terminal costs $c_T(s)$ non-negative. When zero, terminal states are called *goals*. The models underlying 2-player games such as Chess can be understood also in these terms with opponent moves modeled as non-deterministic transitions. Often models are described in terms of rewards rather than costs, or in terms of both, yet care needs to be taken so that models have well-defined solutions. State models of this type are also considered in Control Theory [Bertsekas, 1995], Reinforcement Learning [Sutton and Barto, 1998], and Behavioral Ecology [Houston and McNamara, 1988; Clark, 1991]. In [Astrom, 1965], it is shown how problems involving partial feedback can be reformulated as problems involving full state feedback over *belief states*; i.e., states that

encode the information about the true state of the system. All these problems can also be cast as *search problems* in either the original state space or belief space [Bonet and Geffner, 2000].

The solutions to these various state models have a mathematical form that depends on the type of feedback. In problems without feedback, solutions are sequences of actions, while in problems with full-state feedback solutions are functions mapping states into actions (called also closed-loop control policies). The form of the solution to the various models need to be distinguished from the way they are represented. A common, compact representation of policies is in terms of condition, action rules; yet many of the standard algorithms assume a representation of policies in terms of less-compact value functions. The problem of combining robust algorithms with compact representations is not yet solved, although significant progress has been achieved when actions can be assumed to be deterministic.

From a complexity point of view, if there are n variables, the state space (range of possible value assignments) is exponential in n. Thus, except for problems involving very few variables, exhaustive approaches for specifying or solving these models are unfeasible. A key characteristic of AI Planning are the languages for representing these models, and the techniques used for solving them.

3 Languages

A standard language for representing state models in compact form is Strips [Fikes and Nilsson, 1971].[1] In Strips, a problem P is expressed as a tuple $P = \langle A, O, I, G\rangle$ where A is the set of atoms or boolean variables of interest, O is the set of actions, and $I \subseteq A$, and $G \subseteq A$ are the atoms that are true in the initial and goal situations respectively. In addition, each action $a \in O$ is characterized by three sets of atoms: the atoms $pre(a)$ that must be true in order for the action to be executable (preconditions), the atoms $add(a)$ that become true after the action is done (add list), and finally, the atoms $del(a)$ that become false after doing the action (delete list).

A Strips planner is a *general problem solver* that accepts descriptions of arbitrary problems in Strips, and computes a solution for them; namely, sequences of actions mapping the initial situation into the goal. Actually, any deterministic state model can be expressed in Strips, and any Strips problem $P = \langle A, O, I, G\rangle$ defines a precise state model $S(P)$ where

- the states s are the different subsets of atoms in A
- the initial state s_0 is I
- the goal states s_G are those for which $G \subseteq s_G$
- $A(s)$ is the subset of actions $a \in O$ s.t. $pre(a) \subseteq s$
- $F(a, s) = \{s + Add(a) - Del(a)\}$, for $a \in A(s)$
- the actions costs $c(a, s)$ are uniform (e.g., 1)

Extensions of the Strips language for accommodating non-boolean variables and other features have been developed, and planners capable of solving large and complex problems currently exist. This is the result of new ideas and a solid empirical methodology in AI Planning following [Penberthy and Weld, 1992], [Blum and Furst, 1995], and others in the 90's.

4 Is Strips Planning relevant at all?

Before getting into the techniques that made this progress possible, let us address some common misconceptions about Strips planning. First, it is often said that Strips planning cannot deal with uncertainty. This is true in one way, but not in another. Namely, the model $S(P)$ implicit in a Strips encoding P does not *represent* uncertainty. Yet this does not imply that Strips planning cannot *deal* with uncertainty. It actually can. Indeed, the 'winner' of the ICAPS 2004 Probabistic Planning Competition [Littman, 2005], FF-Replan,[2] is based on a Strips planner called FF [Hoffmann and Nebel, 2001]. While the actions in the domain were probabilistic, FF-Replan ignores the probabilities and replans from scratch using FF after every step. Since currently, this can be done extremely fast even in domains with hundred of actions and variables, this deterministic re-planner did better than more sophisticated probabilistic planners. It does not take much to see that this strategy may work well in a 'noisy' Block Worlds domain where blocks may accidentally fall off gripper, and actually it is not trivial to come up with domains where this strategy will not work (this was indeed the problem in the competition). Control engineers know this very well: stochastic systems are often controlled by closed-loop control policies designed under deterministic approximations, as in many cases errors in the model can be safely corrected through the feedback loop.

A second misconception about Strips or 'classical' planning is that actions denote 'primitive operations' that all take a unit of time. This is not so: Strips planning is about planning with operators that can be chacterized in terms of pre and postconditions. The operator themselves can be abstractions of lower level policies, dealing with low level actions and sensors. For example, the action of grabbing a cup involves moving the arm in certain ways, sensing it, and so on; yet for higher levels, it is natural to assume that the action can be summarized in terms of preconditions involving the proximity of the cup, a free-hand, etc; and postconditions involving the cup in the hand and so on. Reinforcement learning has been shown to be a powerful approach for learning low-level skills, but it has been less successful for integrating these skills for achieving high-level goals. The computational success of Strips planning suggests that one way of doing this is by characterizing low-level behaviors in terms of pre and postconditions, and feeding such behaviors into a planner.

5 Heuristic Search

How can current Strips planners assemble dynamically and effectively low-level behaviors, expressed in terms of pre and post conditions, for achieving goals? The idea is simple: they exploit the structure of the problems by extracting automatically informative heuristic functions. While the idea of using

[1] Strips is the name of a planner developed in the late 60's at SRI, a successor of Newell's and Simon's GPS.

[2] FF-Replan was developed by SungWook Yoon, Alan Fern and Robert Givan from Purdue.

heuristic functions for guiding the search is old [Hart *et al.*, 1968], the idea of extracting these functions automatically from problem encodings is more recent [McDermott, 1996; Bonet *et al.*, 1997], and underlies most current planners.

In order to illustrate the power of heuristic functions for guiding the search, consider the problem of looking in a map for the shortest route between Los Angeles and New York. One of the best known algorithms for finding shortest routes is Dijkstra's algorithm [Cormen *et al.*, 1989]: the algorithm efficiently and recursively computes the shortest distances $g(s)$ between the origin and the closest 'unvisited' cities s til the target is reached. A characteristic of the algorithm when applied to our problem, is that it would first find a shortest path from LA to Mexico City, even if Mexico City is way out of the best path from LA to NY. Of course, this is not the way people find routes in a map. The *heuristic search algorithms* developed in AI approach this problem in a different way, taking into account an estimate $h(s)$ of the cost (distance) to go from s to the goal. In route finding, this estimate is given by the Euclidian distance in the map that separates s from the goal. Using then the sum of the cost $g(s)$ to get to s and an estimate $h(s)$ of the cost-to-go from s to the goal, heuristic or informed search algorithms are much more *focused* than blind search algorithms like Dijkstra, without sacrificing optimality. For example, in finding a route from Los Angeles to New York, heuristic search algorithms like A* or IDA* [Pearl, 1983; Russell and Norvig, 1994], would never consider 'cities' whose value $g(s) + h(s)$ is above the cost of the problem. These algorithms guarantee also that the solutions found are optimal provided that the heuristic function h is *admissible* or *optimistic*, i.e., if for any s, $h(s) \leq V^*(s)$, where V^* is the optimal cost function. In the most informed case, when $h = V^*$, heuristic search algorithms are completely focused and consider only states along optimal paths, while in the other extreme, if $h = 0$, they consider as many states as Dijkstra's algorithm. Most often, we are not in either extreme, yet good informed heuristics can be found that reduce the space to search quite drastically. For example, while with today's technology it is possible to explore in the order of 10^{10} states, optimal solutions to arbitrary configuration of the Rubik's Cube with more than 10^{20} states, have been reported [Korf, 1998]. These search methods are very selective and consider a tiny fraction of the state space only, smaller actually than $1/10^{10}$.

6 Deriving Heuristic Functions

Two key questions arise: 1) How can these heuristics be obtained? and 2) Whether similar gains can be obtained in other models, e.g., when actions are not deterministic and states are not necessarily fully observable. We address each question in turn.

The power of current planners arises from methods for extracting heuristic values $h(s)$ automatically from problem encodings. The idea is to set the estimated costs $h(s)$ of reaching the goal from s to the cost of solving a simpler, relaxed problem. Strips problems, for example, can be relaxed by dropping the delete lists. Solving (non-optimally) a delete-free Strips problem can be done quite efficiently, and the heuristic $h(s)$ can be set to the cost of the relaxation. The idea of obtaining heuristics by solving relaxed problems is old [Pearl, 1983], but the use of Strips relaxations for deriving them automatically for planning is more recent [McDermott, 1996; Bonet *et al.*, 1997]. Since then other relaxations have been been considered. In [Bonet *et al.*, 1997], the derived heuristics are used for selecting actions greedily, in real-time, without finding a complete plan first. The proposal is closely related to the *spreading activation model* of action selection in [Maes, 1990], with *activation levels* replaced by or interpreted as *heuristic values* (cost estimators).

The automatic derivation of heuristic functions for guiding the search provides what is probably the first fast and robust mechanism for carring out means-end analysis in complex domains.

7 Greedy Selection and Lookahead

Heuristic functions, as cost estimators, have also been found crucial for focusing the search in problems involving uncertainty and feedback where solutions are not 'paths' in the state space. Solutions to the various models can be all expressed in terms of control policies π that are *greedy* with respect to a given heuristic function h. A control policy π is a function mapping states $s \in S$ into actions $a \in A(s)$, and a policy π_h is greedy with respect to h iff π_h is the best policy assuming that the cost-to-go is given by h, i.e.

$$\pi_h(s) \;=\; \operatorname*{argmin}_{a \in A(s)} Q_h(a, s) \qquad (1)$$

where $Q_h(a, s)$ is the expression of the cost-to-go whose actual form depends on the model; e.g., for non-deterministic models is $c(a, s) + \max_{s' \in F(a,s)} h(s')$, for MDPs $c(a, s) + \sum_{s' \in F(a,s)} P_a(s'|s)h(s')$, etc. In all cases, if the heuristic h is optimal; i.e., $h = V^*$, the greedy policy π_h is optimal as well [Bellman, 1957; Bertsekas, 1995]. As mentioned above, the planner that won that the last Probabilistic Planning Competition, used a greedy policy based on an heuristic function derived ignoring probabilistic information.

Often, if the heuristic estimator h is good, the greedy policy π_h based on it is good as well. Otherwise, there are two ways for improving the policy π_h without having to consider the entire state space: one is by *look ahead*, the other is by *learning*, and both involve *search*. Look-ahead is the strategy used in 2-player games like Chess that cannot be solved up to the terminal states; it is a variation of the greedy strategy π_h where the $Q_h(a, s)$ term is obtained not from the direct successors of s but from further descendants. The lookahead search is not exhaustive either, as values $h(s')$ of the tip nodes are used to prune the set of nodes considered; a technique known as alpha-beta search [Newell *et al.*, 1963]. The quality of the play depends on the search horizon and on the quality of the value function, which in this case, does not estimate cost but reward. In all the models, the greedy policy π_h is invariant to certain types of transformation in h; e.g $\pi_h = \pi_{h'}$ if $h' = \alpha h + \beta$ for constants α and β, $\alpha > 0$, so the value scale is not critical. Moreover, in Chess, any transformation of the heuristic function that preserves the relative ordering

of the states, yields an equivalent policy, even if lookahead is used.

8 Learning

The second way to improve a greedy policy π_h is by adjusting the heuristic values during the search [Korf, 1990; Barto *et al.*, 1995]. More precisely, after applying the greedy action $\mathrm{argmin}_{a \in A(s)} Q_h(a, s)$ in state s, the heuristic value $h(s)$ in s is updated to

$$h(s) := \min_{a \in A(s)} Q_h(a, s) \quad (2)$$

Interestingly, if h is admissible ($h \leq V^*$), and these updates are performed as the greedy policy π_h is simulated, the resulting algorithm exhibits two properties that distinguish it from standard methods: first, unlike a fixed greedy policy, it will never get trapped into a loop and will eventually get to the goal (if the goal is reachable from every state), and second, after repeated trials, the greedy policy π_h converges to an optimal policy, and the values $h(s)$ to the optimal values $V^*(s)$ (over the relevant states). This algorithm is called Real-Time Dynamic Programming (RTDP) in [Barto *et al.*, 1995] as it combines a greedy, real-time action selecting mechanism, with the improvements brought about by the updates. Like heuristic search algorithms in AI but unlike standard DP methods, RTDP can solve large problems involving uncertainty, without having to consider the whole state space, provided that a good and admissible heuristic function h is used. Moreover, partial feedback can be accommodated as well, by performing the search in 'belief space'. GPT is a planner, that accepts description of problems involving stochastic actions and sensors, and computes optimal or approximate optimal policies using a refinement of these methods [Bonet and Geffner, 2000; 2003].

9 Inference

Many problems have a low polynomial complexity, and are easy for people to solve; e.g., the problem of collecting packages at various destinations in a city, and delivering them at some other destinations. This 'problem' is not even considered a problem by people as, unlike puzzles, can be solved (non-optimally) in a very straightforward way. Yet if the problem is fed to a planner by describing the actions of driving the truck from one location to another, picking up and loading the packages, and so on, the planner would tackle the problem in the same way it would tackle a puzzle: by means of *search*. This search can often be done quite fast, yet like in Chess, this does not seem to be the way people solve such problems. Psychologists interested in problem solving, have focused on puzzles like Towers-of-Hanoi rather than on the simple problems that people solve every day. The work in planning however reveals that problems that are easy for people are not necessarily easy for a general automated problem solver. It may be argued that people solve these problems by using domain-specific knowledge, yet this pushes the problem one level up: how do people recognize when a problem falls in a domain, and how many domains are there? Recently we have addressed the related question of whether it is possible to solve a wide variety of 'simple' problems that are used as benchmarks in planning (including the famous Blocks World problems), by performing efficient (low polynomial) inference and *no search*. To our surprise, we have found that this is possible [Vidal and Geffner, 2005]. We believe that there are a number of useful consequences to draw from this fact, given that most problems faced by real intelligent agents are not puzzles. In any case, inference and heuristic functions are two sides of the same coin: they both extract useful knowledge from a domain description and use it to focus the search, and if possible, to eliminate the search altogether.

10 SAT: Search and Inference in Logic

Logic has played a prominent role in AI as a basis for knowledge representation and programming languages. In recent years, logic restricted to propositional languages has become a powerful computational paradigm as well. A variety of problems can be encoded as SAT problems which are then fed and solved by powerful SAT solvers: programs that take a set of clauses (disjunctions of positive or negated atoms), and determine if the clauses are consistent, and if so, return a truth-valuation that satisfies all the clauses (a model). While the SAT problem is intractable, problems involving thousands of clauses and variables can now be solved [Kautz and Selman, 2005]. Classical planning problems can be mapped into SAT by translating the problem descriptions into propositional logic, and fixing a planning horizon: if the theory is inconsistent, the problem has no solution within the horizon, else a plan can be read off the model [Kautz and Selman, 1996]. For problems involving non-determinism, the SAT formulation yields only 'optimistic' plans, yet work is underway for reproducing the practical success of SAT in richer settings. Current SAT algorithms combine search and inference as well, and are complete. Some of the original algorithms, were based on local search [Selman *et al.*, 1992], and were inspired by a neural-network constraint satisfaction engine [Adorf and Johnston, 1990].

11 Domain Compilation

Another recent development in logic relevant for action selection is *knowledge compilation* [Selman and Kautz, 1996; Darwiche and Marquis, 2002b]. In knowledge compilation, a formula is mapped into a logically equivalent formula of a certain form that makes certain class of operations more efficient. For example, while testing consistency of a formula is exponential in the size of the formula (in the worst case), formulas in d-DNNF can be tested in linear time (d-DNNF is a variation of 'Disjunctive Normal Form'; see [Darwiche, 2001; 2002]). Moreover, for a formula T in d-DNNF, it is possible, in linear-time as well (i.e., very efficiently) to check the consistency of $T + L$ for any set of literals L, get a model of $T + L$, and even count the number of such models. Of course, compiling a formula into d-DNNF is expensive, but this expense is worth if the result of the compilation is used many times. The idea of theory compilation has a number of applications in planning that are beginning to get explored. For example, Barret in [Barret, 2004], compiles planning theories with a fixed planning horizon n

into d-DNNF, and shows that from the compiled theory *it is possible to obtain plans for arbitrary initial situations and goals, in linear-time with no search.* This is a very interesting idea that makes technical sense of the intuition that there are many logically equivalent representations, and yet some representations that are better adapted for a given task. We are currently exploring a variation of Barret's idea that exploits another property of d-DNNF formulas T: the ability to efficiently compute not only models of T but also *best* models, 'best' defined in terms of a ranking over the individual (boolean) variables [Darwiche and Marquis, 2002a]. By ranking the literals at the horizon n and then using ideas similar to ones above, it is possible to get in-linear time, for any initial situation, the best plan and the rank of the (best) situation that it leads to. In this way, very quickly, *we get an appraisal and appropriate 'reaction' to any situation, without doing any search.* It does not take much to relate these appraisals with the role *emotions* in the selection of actions as postulated in a number of recent works; e.g., [Damasio, 1995; Ketelaar and Todd, 2001; Belavkin, 2001; Evans, 2002; MacLeod, 2002]. In the view that arises from domain compilation, however, emotions are prior to search, and they are not used for guiding the search or deliberation, nor are they the result of deliberation; rather they summarize expected reward or penalty (as when a deer sees a lion nearby). This view can account also for the way in which local preferences (ranks) are quickly assembled to provide an assessment of any situation (see "Feeling and Thinking: Preferences Need No Inferences" in [Zajonc, 2004]). Computationally, the account is limited in two ways: it assumes a given fixed planning horizon, and that there is no uncertainty. Still it appears as a good starting point. In relation to the heuristic view of emotions, the notion of domain compilation provides an alternative and probably complementary view: in one case, emotion like an heuristic, guides the search for best reward, in the other, emotion stands for expected reward, which in the compiled representation is computed in linear-time (i.e., very quickly).

12 Summary

We have argued that humans encounter a huge variety of problems which they must solve using general methods. For general methods to work, however, they must be able to recognize and exploit structure. We have then reviewed some recent techniques from AI planning and problem solving that accomplish this, either focusing the search for solutions or bypassing the search altogether. These techniques include the automatic derivation of heuristic functions, the use of limited but effective forms of inference, and the compilation of domains, all of which enable a general problem solver to 'adapt' to the task at hand. We have also discussed briefly how these ideas relate to some biologically-motivated action selection models based on 'activation levels' and recent proposals linking emotions and search.

The area of planning and problem solving in Artificial Intelligence has come a long way, and it is probably time, following Marr's approach, to use the insights gained by the study of *what* is to be computed and *how* is to be computed, for gaining a better understanding of what real-brains actually compute when making plans and selecting actions. Of course, there is a lot to be learned, and many other useful and necessary approaches to the problem, yet some of us hope that a good theory of AI planning and problem solving, as Newell, Simon, and others envisioned many years ago, will be an essential part of the global picture.

References

[Adorf and Johnston, 1990] H. M. Adorf and M. D. Johnston. A discrete stochastic neural network algorithm for constraint satisfaction problems. In *Proc. the Int. Joint Conf. on Neural Networks*, 1990.

[Astrom, 1965] K. Astrom. Optimal control of markov decision processes with incomplete state estimation. *J. Math. Anal. Appl.*, 10:174–205, 1965.

[Barret, 2004] T. Barret. From hybrid systems to universal plans via domain compilation. In *Proc. ICAPS-04*, 2004.

[Barto *et al.*, 1995] A. Barto, S. Bradtke, and S. Singh. Learning to act using real-time dynamic programming. *Artificial Intelligence*, 72:81–138, 1995.

[Belavkin, 2001] R. V. Belavkin. The role of emotion in problem solving. In *Proc. of the AISB'01 Symposium on Emotion, Cognition and Affective Computing*, pages 49–57, 2001.

[Bellman, 1957] R. Bellman. *Dynamic Programming*. Princeton University Press, 1957.

[Bertsekas, 1995] D. Bertsekas. *Dynamic Programming and Optimal Control, Vols 1 and 2.* Athena Scientific, 1995.

[Blum and Furst, 1995] A. Blum and M. Furst. Fast planning through planning graph analysis. In *Proceedings of IJCAI-95*, pages 1636–1642. Morgan Kaufmann, 1995.

[Bonet and Geffner, 2000] B. Bonet and H. Geffner. Planning with incomplete information as heuristic search in belief space. In *Proc. of AIPS-2000*, pages 52–61. AAAI Press, 2000.

[Bonet and Geffner, 2001] B. Bonet and H. Geffner. Planning as heuristic search. *Artificial Intelligence*, 129(1–2):5–33, 2001.

[Bonet and Geffner, 2003] B. Bonet and H. Geffner. Labeled RTDP: Improving the convergence of real-time dynamic programming. In *Proc. 13th Int. Conf. on Automated Planning and Scheduling (ICAPS-2003)*, pages 12–31. AAAI Press, 2003.

[Bonet *et al.*, 1997] B. Bonet, G. Loerincs, and H. Geffner. A robust and fast action selection mechanism for planning. In *Proceedings of AAAI-97*, pages 714–719. MIT Press, 1997.

[Brooks, 1997] R. Brooks. From earwigs to humans. *Robotics and Autonomous Systems*, 20(2-4):291–304, 1997.

[Clark, 1991] C. Clark. Modeling behavioral adaptations. *Behavioral and Brain Sciences*, 14(1), 1991.

[Cormen *et al.*, 1989] T. H. Cormen, C. E. Leiserson, and R. L. Rivest. *Introduction to Algorithms*. The MIT Press, 1989.

[Damasio, 1995] A. Damasio. *Descartes' Error: Emotion, Reason, and the Human Brain*. Quill, 1995.

[Darwiche and Marquis, 2002a] A. Darwiche and P. Marquis. Compilation of propositional weighted bases. In *Proc. NMR-02*, 2002.

[Darwiche and Marquis, 2002b] A. Darwiche and P. Marquis. A knowledge compilation map. *J. of AI Research*, 17:229–264, 2002.

[Darwiche, 2001] Adnan Darwiche. Decomposable negation normal form. *J. ACM*, 48(4):608–647, 2001.

[Darwiche, 2002] A. Darwiche. On the tractable counting of theory models and its applications to belief revision and truth maintenance. *J. of Applied Non-Classical Logics*, 2002.

[Evans, 2002] D. Evans. The search hypothesis of emotion. *British J. Phil. Science*, 53:497–509, 2002.

[Fikes and Nilsson, 1971] R. Fikes and N. Nilsson. STRIPS: A new approach to the application of theorem proving to problem solving. *Artificial Intelligence*, 1:27–120, 1971.

[Gigerenzer and Todd, 1999] G. Gigerenzer and P. Todd. *Simple Heuristics that Make Us Smart*. Oxford, 1999.

[Glimcher, 2003] P. Glimcher. *Decisions, Uncertainty, and the Brain: The Science of Neuroeconomics*. MIT Press, 2003.

[Hart *et al.*, 1968] P. Hart, N. Nilsson, and B. Raphael. A formal basis for the heuristic determination of minimum cost paths. *IEEE Trans. Syst. Sci. Cybern.*, 4:100–107, 1968.

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

[Houston and McNamara, 1988] A. I. Houston and J. M. McNamara. A framework for the functional analysis of behaviour. *Behavioral and Brain Sciences*, 11(1), 1988.

[Kautz and Selman, 1996] H. Kautz and B. Selman. Pushing the envelope: Planning, propositional logic, and stochastic search. In *Proceedings of AAAI-96*, pages 1194–1201. AAAI Press / MIT Press, 1996.

[Kautz and Selman, 2005] H. Kautz and B. Selman. The state of SAT. *Discrete and Applied Math*, 2005.

[Ketelaar and Todd, 2001] T. Ketelaar and P. M. Todd. Framing our thoughts: Evolutionary psychology's answer to the computational mind's dilemma. In H.R. Holcomb III, editor, *Conceptual Challenges in Evolutionary Psychology*. Kluwer, 2001.

[Korf, 1990] R. Korf. Real-time heuristic search. *Artificial Intelligence*, 42:189–211, 1990.

[Korf, 1998] R. Korf. Finding optimal solutions to Rubik's cube using pattern databases. In *Proceedings of AAAI-98*, pages 1202–1207. AAAI Press / MIT Press, 1998.

[Littman, 2005] Michael Littman. The first probabilistic planning competition. To be published, 2005.

[MacLeod, 2002] W. Bentley MacLeod. Complexity, bounded rationality and heuristic search. *Contributions to Economic Analysis & Policy*, 1(1), 2002.

[Maes, 1990] P. Maes. Situated agents can have goals. *Robotics and Autonomous Systems*, 6:49–70, 1990.

[McDermott, 1996] D. McDermott. A heuristic estimator for means-ends analysis in planning. In *Proc. Third Int. Conf. on AI Planning Systems (AIPS-96)*, 1996.

[McFarland and Bosser, 1993] David McFarland and Thomas Bosser. *Intelligent behaviour in animals and robots*. MIT Press, 1993.

[Newell and Simon, 1958] A. Newell and H. Simon. Elements of a theory of human problem solving. *Psychology Review*, 1958.

[Newell and Simon, 1963] A. Newell and H. Simon. GPS: a program that simulates human thought. In E. Feigenbaum and J. Feldman, editors, *Computers and Thought*, pages 279–293. McGraw Hill, 1963.

[Newell *et al.*, 1963] A. Newell, J. C. Shaw, and H. Simon. Chess-playing programs and the problem of complexity. In E. Feigenbaum and J. Feldman, editors, *Computers and Thought*, pages 109–133. McGraw Hill, 1963.

[Pearl, 1983] J. Pearl. *Heuristics*. Addison Wesley, 1983.

[Penberthy and Weld, 1992] J. Penberthy and D. Weld. Ucpop: A sound, complete, partiall order planner for adl. In *Proceedings KR'92*, 1992.

[Russell and Norvig, 1994] S. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach*. Prentice Hall, 1994.

[Selman and Kautz, 1996] Bart Selman and Henry Kautz. Knowledge compilation and theory approximation. *Journal of the ACM*, 43(2):193–224, 1996.

[Selman *et al.*, 1992] B. Selman, H. Levesque, and D. Mitchell. A new method for solving hard satisfiability problems. In *Proc. AAAI-92*, 1992.

[Smith, 2003] D. Smith. Special issue on the 3rd international planning competition. *JAIR*, 20, 2003.

[Stanovich, 2004] K. Stanovich. *The Robot's Rebellion: Finding Meaning in the Age of Darwin*. Chicago, 2004.

[Sutton and Barto, 1998] R. Sutton and A. Barto. *Introduction to Reinforcement Learning*. MIT Press, 1998.

[Tooby and Cosmides, 1992] J. Tooby and L. Cosmides. The psychological foundations of culture. In J. Barkow, L. Cosmides, and J. Tooby, editors, *The Adapted Mind*. Oxford, 1992.

[Vidal and Geffner, 2005] V. Vidal and H. Geffner. Solving simple planning problems with more inference and no search. 2005.

[Zajonc, 2004] R. Zajonc. *The Selected Works of R.B. Zajonc*. Wiley, 2004.