Mind Soc (2014) 13:83–95
DOI 10.1007/s11299-014-0143-y

Bounded rationality in problem solving: Guiding search with domain-independent heuristics

Pat Langley · Chris Pearce · Mike Barley · Miranda Emery

Received: 20 February 2014 / Accepted: 21 March 2014 / Published online: 23 April 2014

Abstract Humans exhibit the remarkable ability to solve complex, multi-step problems despite their limited capacity for search. We review the standard theory of problem solving, which posits that heuristic guidance makes this possible, but we also note that most studies have emphasized the role of domain-specific heuristics, which are not available for unfamiliar tasks, over more general ones. We describe FPS, a flexible architecture for problem solving that supports a variety of different strategies and heuristics, and we report its use in an experiment that studies the effectiveness of two domain-independent criteria. The results suggest that such heuristics can make problem solving far more tractable, and they are generally consistent with our claim that their use offsets the drawbacks of bounded rationality.

Keywords Problem solving · Heuristic search · Bounded rationality

1 Introduction and motivation

The ability to solve novel problems is a hallmark of human cognition, being one of the key features that distinguishes our species from other animals. By 'problem solving' we refer to tasks that require one to carry out a sequence of steps to achieve some objective and that require one to make choices for at least some of these steps. Analysis of such multi-step tasks suggests that finding solutions often involves *search* through a problem space that grows exponentially with length of the solution path.

Yet studies of human cognition have shown than our rationality is bounded even in settings that involve only simple choice among a set of alternatives. These limits

P. Langley (✉) · C. Pearce · M. Barley · M. Emery
Department of Computer Science, University of Auckland, Private Bag 92019, Auckland 1142, New Zealand
e-mail: patrick.w.langley@gmail.com

become still more important in the context of multi-step problem solving, but people remain able to solve many tasks of this sort despite their limited information-processing abilities. The most common explanation is that humans rely on *heuristics* to limit or guide their problem solving. Heuristics are criteria or rules of thumb that, although not guaranteed to produce the best decision, often suggest good ones that help an agent achieve his goals. In multi-step problem solving, heuristics can reduce substantially the effective branching factor of search, making tasks tractable that would otherwise fall beyond the scope of human abilities.

However, most research on heuristics in problem solving has emphasized the role of *domain-specific* knowledge in reducing search. This idea has dominated both psychological studies of problem solving in domains like physics (Larkin et al. 1980) and computational models of this ability (e.g., Carbonell et al. 1990; Laird et al. 1987). Such domain-specific heuristics can be powerful, but they are not available when a person first encounters a problem in a new domain. Nevertheless, in many cases novices can solve such unfamiliar problems, albeit more haltingly and with more search than in familiar settings. How can we reconcile this finding with bounded rationality?

In this paper, we propose that human behavior on unfamiliar tasks is effective because it takes advantage of *general* heuristics that hold across many different domains.[1] In the next section, we review the standard theory of problem solving, including the central role played by heuristic search. After this, we describe the Flexible Problem Solver, a computational account of problem solving that attempts to explain the variability seen in humans. Next we report a computational experiment designed to clarify the benefits of domain-independent heuristics in a number of problem-solving domains. In closing, we discuss related work on the topic and consider some directions for future research.

2 The standard theory of problem solving

Langley and Rogers (2005) have reviewed the most widely accepted account of human problem solving—originally proposed by Newell et al. (1958)—which we will refer to as the *standard theory*. This characterizes, in computational terms, how people solve unfamiliar problems that require them to find a sequence of steps to achieve some desired state. Their early studies focused on abstract problems like the Tower of Hanoi puzzle and proving theorems in logic, but they later extended it to cover semantically rich domains like physics and thermodynamics.

The standard theory of problem solving makes a number of clear claims, of increasing specificity, that are worth recounting here. The first is a very general statement about the character of high-level human cognition. The theory posits that this depends on the representation, interpretation, and manipulation of *symbol structures*, that is, organized structures of entities that denote some situation or activity and that appear in some form that persists over time. Newell and Simon

[1] These are similar in spirit to heuristics studied by Gigerenzer and Todd (1999), but they guide search during multi-step problem solving rather than choice on simple decision-making tasks.

(1976) later refined this idea into their *physical symbol system* hypothesis, which stated that the representation and processing of symbol structures are both necessary and sufficient for general intelligent behavior.

However, this assumption applies to all aspects of high-level cognition, including language understanding and routine execution of complex skills. What distinguishes problem solving on unfamiliar tasks is the need to make a sequence of choices that, taken together, achieve the agent's goal or objective. The standard theory states that humans respond by carrying out *search through a problem space*. They encode the candidate *states* in this space as symbol structures that denote different situations that can arise, and they generate these states from others (starting from an initial state) by applying *operators*. Each operator specifies some action, how it transforms the current state into a new one, and the conditions under which these effects occur. Most operators are generic, in that they can apply to different states, with their applicability being tested through *pattern matching*. This process involves comparing one symbol structure, say a general pattern like the conditions on an operator, to another structure, say a more specific one like a concrete state. The problem solver also uses pattern matching to determine whether a generated state satisfies a goal description.

Taken together, an initial problem state and a set of operators implicitly define the problem space. However, this space often has a combinatorial character, in that the number of states grows exponentially with the number of operators applied. Although exhaustive search is possible for sufficiently simple problems, it is impractical for most tasks. In response, the standard theory's third assumption is that humans use *heuristics* to guide selective search through the problem space. Such rules of thumb are not guaranteed to recommend the best choice, or even a good one, but they do so often enough that they make search through combinatorial spaces tractable. Newell and Simon (1976) referred to this claim as the *heuristic search* hypothesis.

Moreover, human problem solvers often utilize a strategy that Newell et al. (1960) referred to as *means-ends analysis*. This technique finds differences between the current problem state and the desired one, then retrieves operators that, if applied, would reduce one or more of the differences. Once the mechanism has selected an operator from among these candidates, it either applies the operator (if its conditions are satisfied) or generates a subtask to make it applicable. Applying an operator produces a new state that either matches the desired state or, if not, leads to another round of finding differences and selecting an operator to apply. Humans do not always rely on means-ends analysis, but it appears frequently enough to merit inclusion in the list of tenets.

Empirical studies of human problem solving (e.g., Newell and Simon 1972) have repeatedly found evidence that supports this standard theory, and there have been no serious critiques of its adequacy. This makes it one of the most robust and stable contributions to our understanding of high-level cognition. However, this does not mean the theory is complete, in the sense that it does not account for all facets of problem solving. For instance, it does not explain the initial construction of problem spaces (e.g., Hayes and Simon 1974) or the role of insight in certain kinds of tasks

(Ohlsson 1992), so there remain areas in which we must elaborate on it further, as Langley and Rogers (2005) have attempted.

One topic especially deserving of attention is the role of general heuristics that apply across many domains. As mentioned earlier, detailed studies of heuristics have typically focused on ones that incorporate domain knowledge. In particular, two major computational architectures that elaborate on the standard theory, Soar (Laird et al. 1987) and Prodigy (Carbonell et al. 1990), both emphasize the use of domain-specific heuristics that are acquired through problem-solving experience. There seems no question that such heuristics play an important role in expert behavior, but they do not explain the human ability to solve unfamiliar tasks without such knowledge. In the remainder of this paper, we examine two general heuristics that may explain this facility. However, first we must describe the computational framework in which we will embed them.

3 The FPS architecture

We have chosen to study domain-independent heuristics in the context of FPS, an architecture for flexible problem solving (Langley et al. 2013). We will not claim that FPS is superior to similar frameworks like Soar and Prodigy, but we designed it to account for the great variety observed in human behavior, including their use of different search strategies and choice criteria. Thus, it seems well suited to studies of how general heuristics can guide problem solving even in the absence of domain knowledge. In this section, we review briefly the architecture's representations and processes.

3.1 Representational assumptions

Like many accounts of human cognition, the FPS architecture partitions symbolic mental structures into a *working memory* that contains dynamic elements and a *long-term memory* that contains stable knowledge. The former changes rapidly over the course of problem solving, as new structures are introduced. The latter changes only through learning, which occurs gradually and which does not concern us here.

The central structure in FPS' working memory is the *problem*, which includes a state description and a goal description. Both are described by a set of elements which share a common symbol that denote the state and goal names, respectively. A given problem also has a unique identifier that distinguishes it from other problems. Another important structure is an *intention*, which specifies an instance of some operator that the agent intends to apply. An applied intention I decomposes its parent problem P into two subtasks: a *down* subproblem with the same state as P but with goals based on I's conditions, and a *right* subproblem with the same goal description as P but with a state created by applying the intention to P.

FPS encodes a *problem solution* as a set of connected elements in working memory. A problem P and its applied intention I is a direct solution if I has produced a state that matches the goal description associated with P. A more complex problem-intention pair is a solution if both of its associated subproblems are either trivially

solved, in that their states match their goals, or if they have intentions whose application offers solutions. The same problem may have more than one associated intention, but at most one may play part in a given problem solution.

Long-term memory in FPS contains two forms of stable content. The first involves *domain knowledge*, which specifies the predicates used to describe states and goals, the operators that generate new states, and the inference rules that elaborate on their descriptions. This knowledge varies across domains, as these typically involve different types of objects, relations, and activities. The second type, *strategic knowledge*, specifies how the agent should search the problem space and is domain independent. Domain knowledge provides the notation that FPS uses to generate its new intentions, states, and goals. In contrast, strategic knowledge determines the details of the problem-solving process, which we can now describe.

3.2 Problem-solving mechanisms

The FPS architecture approaches problem solving in much the same manner as Newell et al. (1960) General Problem Solver, in that it attempts to transform an initial state into another one that satisfies the goal description. This process involves the selection and application operators that produce new states and, along the way, lead FPS to generate subproblems. The architecture operates in discrete cognitive cycles, each of which involves five stages that use knowledge in long-term memory to create problem-related structures in working memory.

In the first stage, FPS selects a problem P on which to focus its attention, excluding ones that have already been solved or rejected. Depending on which strategic knowledge resides in long-term memory, this phase traverses the problem space in different orders. For instance, the architecture may carry out depth-first search, breadth-first search, or iterative sampling (Langley 1992). The latter, which we assume in this paper, involves greedy selection of intentions, with the system returning to the top-level problem when encountering failure. This strategy is similar to progressive deepening, which de Groot (1978) has observed in human chess players.

During the second stage, the architecture selects an operator instance that appears relevant to solving the current problem P and associates it with P as an intention. Different strategic knowledge for this phase can lead FPS to generate candidate intentions with effects that would achieve at least one of P's goals or ones whose conditions fully match against P's state. For this paper, we will assume the former scheme, which produces the form of means-ends problem solving that plays a key role in Newell et al.'s standard theory.

Heuristics for selecting among intentions come into play during this second stage. Here we will focus on two domain-independent heuristics. The first prefers candidate operator instances that, after their application, lead to more satisfied goals;[2] we will refer to this as the *more goals* heuristic. This bias should reduce the number of steps required to achieve the remaining goals. The second prefers

[2] FPS computes this as a difference between the number of previously unsatisfied goals that would be achieved by the operator and the number of previously satisfied goals that would be undone.

operator instances that have fewer unsatisfied conditions; we will refer to this as the *fewer conditions* heuristic. This bias should reduce the number of steps needed to make the operator applicable and thus achieve its intended goals. When both heuristics are used, they are summed to produce an overall score.

FPS' third stage is responsible for attempting to apply the selected intention. Different knowledge can produce either a delayed commitment strategy or eager commitment, which we assume here. In this case, the architecture attempts to apply an intention I for the current problem P as soon as one has been selected. If its conditions are not satisfied, then FPS generates a down subproblem with the same state as P but with goals based on I's conditions. If the current state meets conditions, then the system generates a right subproblem with the same goals as P but with a state that reflects I's effects. This scheme is very similar to that in Newell et al.'s (1960) General Problem Solver, as well as Carbonell et al.'s (1990) Prodigy.

The architecture's final two stages check for failure and success. Strategic knowledge for detecting failure may include testing for loops (a problem being the same as one of its ancestors) and exceeding a depth limit, both of which we will assume in this paper. Knowledge about success may include tests for a state that satisfies all elements of the goal description, a certain fraction of elements, or some other criterion. Here we will assume the first option, so that FPS views a problem or subproblem as solved only when it reaches a state that achieves all of its goals.

Although we designed FPS with flexibility in mind, we will focus here on a specific variant that combines iterative sampling, goal-driven intention generation, and eager commitment. We believe this combination of strategies, although not the only ones found in human problem solving, arises frequently enough that it provides a solid base from which to study our main interest, the manner in which domain-independent heuristics can make search through a combinatorial space tractable in the presence of bounded rationality.

4 A study of heuristic effectiveness

Our main theoretical claim—that domain-independent heuristics enable solution of novel problems despite the potential for substantial search—is consistent with observations of human cognition, but we would like a deeper understanding of their benefits. To this end, we designed and ran an experiment with the FPS architecture, a convenient platform because its modular character lets us remove, add, and alter components as desired. Our aim was not to match the details of human problem solving, but rather to clarify the ways in which heuristics aid this process. In this sense, our study is similar in spirit to computational experiments by Gigerenzer and Todd (1999) on the benefits of heuristics in simple choice tasks.

We focused on the two generic criteria described earlier, one which prefers intentions that achieve more unsatisfied goals and another that favors candidates with fewer unmatched conditions. We selected five domains for the experiment—the Tower of Hanoi, Logistics, Dock Workers, Blocks World, and N Queens—which we summarize in Table 1. The first is a well-known puzzle that has been studied extensively (e.g., Simon 1975). The next three are similar in character, in

Table 1 Five problem-solving domains used in our experiment on heuristic guidance

Tower of Hanoi. This domain involves *N* disks that sit on three pegs. The single operator can move a disk onto a new peg if there is no smaller disk on it and if there is no smaller disk on the new peg. Goal descriptions specify desired placements of disks on pegs.
Logistics. This domain includes a number of cities, locations in those cities, transportable packages, and trucks and plans that can hold those packages. Operators involve loading and unloading packages, flying planes between cities, and driving trucks between locations. Goals specify desired locations of packages.
Dock Workers. This domain contains a robot, containers, pallets, and cranes in various locations. Operators let the robot move between locations and let the crane transfer a container from the top of a stack to the robot or vice versa. Goal descriptions specify desired positions of containers on pallets in locations.
Blocks World. This domain involves a number of blocks that may be stacked on a table. Any number of piles may be used while solving the problem, and the goal descriptions specify the desired stacking of blocks on top of each other or the table.
N Queens. This domain involves the placement of *N* queen pieces on an *N* by *N* chess board. The goal description states that the queens must be positioned on the board such that no queen may attack another queen. Problems vary in the number of queens and size of the board.

that they use a few abstract operators to define a nontrivial problem space that most people find challenging but still tractable. The last is a constraint-satisfaction task that involves a large state space, but for which the order of operators in a solution does not matter.

Our dependent variables measured different aspects of problem-solving performance. These include, for each problem, the number of intentions that FPS considers during search, the number of subproblems that the system generates, the number of attempts that iterative sampling makes before finding a solution, the number of FPS cycles required to solve the task, and the percentage of problems solved within 10,000 problem-solving cycles. We recorded these metrics for each task provided to the system. When FPS did not find a solution within the allotted cycles, we used the totals when it reached this limit.[3] Our independent variables are the presence or absence of the two domain-independent heuristics.

Together, these specifications let us operationalize our hypothesis: *The use of domain-independent heuristics can make means-ends problem solving, combined with iterative sampling, more effective than random search*. More specifically, they reduce the number of intentions and subproblems generated, increase the chances of finding a solution, and reduce the number of solution attempts required. We are interested both in the individual benefits of the two heuristics and in their joint contribution to reducing search effort.

We tested this hypothesis by generating ten problems of varying difficulty for each of the five domains. Tower of Hanoi problems ranged from three to four disks, Logistics tasks varied from three locations and one package to five locations and three packages, Dock Workers problems varied from moving three containers between two pallets in one location to three containers, two pallets, and two rooms,

[3] Because the heuristic versions of FPS solved more problems in the cycles available, this scheme favors the nonheuristic variant, which provides a more stringent test of our claim.

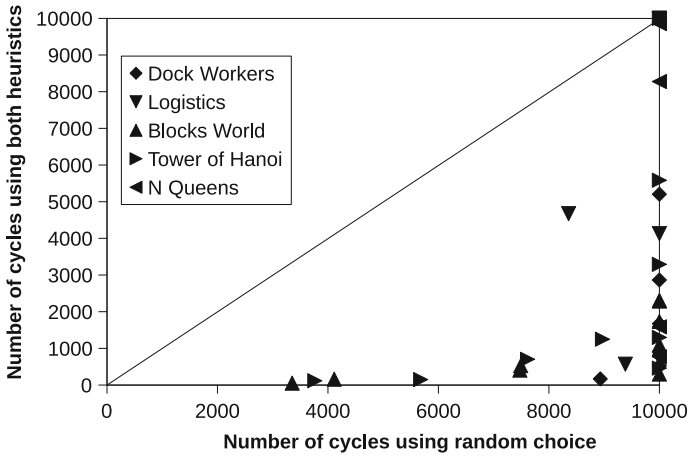

Fig. 1 Average number of cycles taken to solve, or fail on, problems by the nonheuristic FPS and a version that used both the 'fewer conditions' and 'more goals' heuristics

Blocks World problems varied from four to five blocks, and N Queens problems varied from four to ten queen pieces. We ran four distinct variants of FPS: no heuristics, only the 'more goals' heuristic, only the 'fewer conditions' heuristic, and both heuristics. Because the iterative sampling strategy operates stochastically, we ran the system five times for each problem-condition pair and averaged the results.

Figure 1 presents a scatter plot that compares the average number of cycles for the nonheuristic FPS with the version that uses both heuristics. Every point lies below the diagonal, which means that the heuristic version uniformly required less effort to solve problems than the version that selected intentions without this bias. Moreover, the effect is substantial. On average, FPS needed 2.0 times as many FPS cycles without the heuristics as when they were available. Similar results hold for our other dependent measures. The respective factors were 2.1 for the number of intentions, 1.6 for the number of subproblems, and 1.2 for the number of solution attempts. With random selection, FPS succeeded on only 15 % of runs in the cycles available, whereas it found solutions on 64 % when using the two heuristics.

We are also interested in the relative contributions of the two heuristics to problem-solving behavior, so we examined FPS' behavior with each of the heuristics in isolation. The number of cycles used by the 'fewer conditions' variant was similar to that in Fig. 1. On average, the nonheuristic FPS took 1.5 times as many cycles, produced 1.6 as many intentions, and created 1.1 times as many subproblems. One surprise was that it actually made fewer attempts, only 0.6 of the heuristic version, during problem solving. We suspect that this is because unguided search seldom leads less often to loops that cause iterative sampling to restart, but we have not tested this hypothesis. The system found solutions only 15 % of the time in the allotted cycles without the heuristic and succeeded on 50 % of the runs with it.

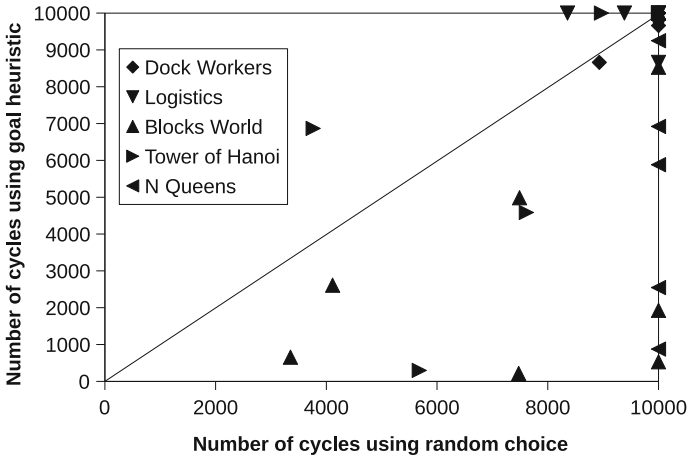

Fig. 2 Average number of cycles taken to solve, or fail on, problems by the nonheuristic FPS and a version that used only the 'more goals' heuristic

The benefits of the 'more goals' heuristic, when used alone, were far less, although still generally positive. Figure 2 presents a second scatter plot that compares the average number of cycles that FPS required, on each problem, with only this heuristic and the nonheuristic variant. In this case, both versions failed to solve some Dock Workers problems, and the heuristic variant even took longer for the simpler Logistics and Tower of Hanoi tasks. Nevertheless, on average the random FPS variant took 1.2 times as many cycles, produced 1.3 times as many subproblems, and generated 1.2 times the number of intentions as the 'more goals' variant. Again, random search actually made fewer attempts, only 0.7 of the heuristic version, during problem solving. Moreover, this heuristic led FPS to find solutions on only 28 % of the problem runs, only a little better than the nonheuristic variant.

From these results, it seems clear that the 'fewer conditions' heuristics accounted for most of the improvement, with the 'more goals' criterion aiding FPS much less on average and in some cases even leading it astray. This is not a major concern, as heuristics do not come with guarantees,[4] but we would still like to understand the difference. A plausible explanation is that this version of FPS emulated means-ends analysis, which only considers operators whose application would achieve at least one of the current problem's goals. Thus, some of the heuristic's potential benefits were already handled by the intention generation strategy. Another factor is that, in all five domains, operators typically achieve only one goal at a time, so that ranking them by the number of goals provides little assistance.

In summary, our experimental results were largely consistent with the hypothesis that domain-independent heuristics can make problem solving tractable on unfamiliar problems. One criterion was more effective for this end than the other,

[4] At least this holds for the initial sense of the term. Work on 'admissable heuristics'—the use of which ensures finding an optimal problem solution—runs counter to the intent of the standard theory.

but we found the same qualitative results along a number of dependent measures that reflect how much effort FPS required to find solutions. This lends support for the claim that such heuristics are responsible for the human ability to solve combinatorial problems despite their bounded mental resources.

5 Related research

The idea of domain-independent heuristics is not a new one, but it has received relatively little attention in the literature. Newell (1969) championed the notion of *weak methods* that have general applicability but require search through a problem space, but he did not focus on criteria for guiding this search. In a recent review, Ohlsson (2012) has remarked on our lack of knowledge about the heuristics people use during problem solving:

> Subjects' initial choices in this and many other problem spaces appear to be shaped by unconscious dispositions that have little rational basis. We understand next to nothing about the origin and operation of such dispositions.

Some computational models of problem solving (Carbonell et al. 1990; Laird et al. 1987; Langley 1985; Ohlsson 1983) have addressed the acquisition of domain-specific heuristics, usually stated as rules that suggest moves to take or avoid, but there has been little effort devoted to domain-independent criteria.

Ohlsson and Langley (1988) proposed a number generic criteria that their DPF system used, not to guide the process of problem solving, but to evaluate candidate solution paths during automated cognitive diagnosis. Most of their heuristics operate over entire solution paths, which means they are not helpful during the search process. However, one criterion involved a preference for paths that make lower demands on short-term memory; this could be used during problem solving and constitutes a generic heuristic of the type we have proposed.

Generic heuristics also appear in the artificial intelligence literature, but sometimes as the topic of abstract discussions rather than as a focus of empirical study. For instance, Ginsberg and Geddis (1991) argue for the usefulness of domain-independent biases such as "working on more highly constrained tasks before less constrained ones". In response, Minton (1996) argues that, although such general criteria can indeed reduce search, the cost of making them operational through inference can offset this advantage. Instead, he favors reliance on domain-specific heuristics that can be applied more efficiently.

Stone et al. (1994) report one empirical study of three domain-independent heuristics on a number of synthetic domains using the Prodigy architecture. They found that one criterion improved problem solving more than others on these tasks, but then constructed another domain on which the ordering was reversed. They concluded by advocating the need to favor different heuristics depending on features of the domain at hand. This left the door open to the use of generic criteria, but it again downplayed generality and emphasized the role of domain knowledge in guiding the search process.

More recent AI work on planning often incorporates two generic heuristics that are based on different forms of abstraction. One approach ignores effects that involve deletion to find an idealized solution, then uses the length or cost of this solution as an evaluation function in the original space (Hoffmann and Nebel 2001). Another ignores certain aspects of states to create a simplified problem, finds a solution in this reduced space, and again uses its cost to guide search in the initial space (Edelkamp 2001; Haslum et al. 2005). However, both techniques can involve substantial search through the abstract space, which diverges from the sense of 'heuristic' that Newell and Simon championed originally.

A few researchers have incorporated domain-independent heuristics into problem solvers that are similar to those we have examined. For example, Jones and Langley (2005) report a means-ends architecture that uses a version of the 'more goals' criterion, while Nejati (personal communication, 2010) has explored the 'fewer conditions' metric. Trivedi and Langley (2013) describe an extension to the ICARUS architecture that can use either heuristic separately, but their framework differs from FPS in that it does not combine their scores into a joint criterion.

6 Concluding remarks

In this paper, we reexamined the standard theory of problem solving, which claims that, despite the bounds on their information-processing abilities, humans can search effectively through combinatorial spaces to find solutions. We reviewed the idea that heuristic guidance makes this possible, and we argued that, when confronted with novel tasks, people resort to domain-independent criteria to direct their search in promising directions. We described FPS, an architecture for problem solving that incorporates the key tenets of the standard theory, and we reported an experiment that tested the benefits of two heuristics on five different domains.

The results were generally consistent with our hypothesis that domain-independent heuristics can make problem solving more effective. We found that the 'fewer conditions' criterion was better at reducing search than the 'more goals' heuristic, but also that their combination led to substantially less effort than problem solving with random operator selection. This clarifies how humans, despite their bounded rationality, can solve novel combinatorial problems without the benefits of domain knowledge. We also saw that, although this idea has been explored in the computational literature, it has not been studied as fully as its importance merits.

Our own research on this topic has only scratched the surface of this central phenomenon. In future work, we should attempt to identify other domain-independent heuristics and study their effectiveness. We should also examine how these criteria interact with problem-solving strategies, such as whether one carries out forward chaining rather than means-ends analysis or utilizes depth-first search rather than iterative sampling. In addition, we should examine traces of human problem solving to determine what percentage of their decisions can be explained by one or more of these heuristics. Despite the fact that Newell et al. introduced the standard theory over 50 years ago, there remain many unanswered questions that deserve attention from the research community.

Acknowledgements This research was supported by Grant No. N00014-10-1-0487 from the Office of Naval Research, which is not responsible for its contents. We thank Jaime Carbonell, Steve Minton, Peter Stone, and Manuela Veloso for providing information about their work on Prodigy, as well as Chris MacLellan, Trevor Gee, and Colin Walker for their efforts on design and implementation of earlier versions of the FPS system. The first author is also affiliated with Carnegie Mellon Silicon Valley and Institute for the Study of Learning and Expertise.

References

Carbonell JG, Knoblock CA, Minton S (1990) Prodigy: An integrated architecture for planning and learning. In: VanLehn K (ed) Architectures for intelligence. Lawrence Erlbaum, Hillsdale

de Groot AD (1978) Thought and choice in chess, 2nd edn. Mouton Publishers, The Hague

Edelkamp S (2001) Planning with pattern databases. Proceedings of the sixth European conference on planning. Toledo, Spain, pp 13–24

Gigerenzer G, Todd PM (1999) Simple heuristics that make us smart. Oxford University Press, New York

Ginsberg ML, Geddis DF (1991) Is there any need for domain-dependent control information? Proceedings of the ninth national conference on artificial intelligence. AAAI Press, Anaheim, CA, pp 452–457

Hayes JR, Simon HA (1974) Understanding written problem instructions. In: Gregg LW (ed) Knowledge and cognition. Lawrence Erlbaum, Potomac

Haslum P, Bonet B, Geffner, H (2005) New admissible heuristics for domain-independent planning. Proceedings of the twentieth national conference on artificial intelligence. AAAI Press, Pittsburgh, pp 1163–1168

Hoffmann J, Nebel B (2001) The FF planning system: fast plan generation through heuristic search. J Artif Intell Res 14:253–302

Jones RM, Langley P (2005) A constrained architecture for learning and problem solving. Comput Intell 21:480–502

Laird JE, Newell A, Rosenbloom PS (1987) Soar: An architecture for general intelligence. Artif Intell 33:1–64

Langley P (1985) Learning to search: from weak methods to domain-specific heuristics. Cogn Sci 9:217–260

Langley P (1992) Systematic and nonsystematic search strategies. Proceedings of the first international conference on artificial intelligence planning systems. Morgan Kaufmann, College Park, MD, pp 145–152

Langley P, Emery M, Barley M, MacLellan CJ (2013) An architecture for flexible problem solving. Poster collection from the second annual conference on advances in cognitive systems. Baltimore, MD, pp 93–110

Langley P, Rogers S (2005) An extended theory of human problem solving. Proceedings of the twenty-seventh annual meeting of the cognitive science society, Stresa, Italy

Larkin JH, McDermott J, Simon DP, Simon HA (1980) Expert and novice performance in solving physics problems. Science 208:1335–1342

Minton S (1996) Is there any need for domain-dependent control information? A reply. Proceedings of the thirteenth national conference on artificial intelligence. AAAI Press, Portland, pp 855–862

Newell A (1969) Heuristic programming: Ill-structured problems. In: Aronofsky J (ed) Progress in operations research (vol III). John Wiley, New York, pp 360–414

Newell A, Shaw JC, Simon HA (1958) Elements of a theory of human problem solving. Psychol Rev 65:151–166

Newell A, Shaw JC, Simon HA (1960) Report on a general problem-solving program for a computer. Proceedings of the international conference on information processing. UNESCO House, Paris, pp 256–264

Newell A, Simon HA (1972) Human problem solving. Prentice-Hall, Englewood Cliffs

Newell A, Simon HA (1976) Computer science as empirical enquiry: Symbols and search. Commun ACM 19:113–126

Ohlsson S (1983) A constrained mechanism for procedural learning. Proceedings of the eighth international joint conference on artificial intelligence. Morgan Kaufmann, Karlsruhe, Germany, pp 426–428

Ohlsson S (1992) Information processing explanations of insight and related phenomena. In: Keane M, Gilhooly K (eds) Advances in the psychology of thinking (Vol. 1). Harvester-Wheatsheaf, London
Ohlsson S (2012) The problems with problem solving: Reflections on the rise, current status, and possible future of a cognitive research paradigm. J Probl Solving 5:101–128
Ohlsson S, Langley P (1988) Psychological evaluation of path hypotheses in cognitive diagnosis. In: Mandl H, Lesgold A (eds) Learning issues for intelligent tutoring systems. Springer, New York, pp 42–62
Simon HA (1975) The functional equivalence of problem solving skills. Cogn Psychol 7:268–288
Stone P, Veloso MM, Blythe J (1994) The need for different domain-independent heuristics. Proceedings of the second international conference on AI planning systems. AAAI Press, Chicago, pp 164–169
Trivedi N, Langley P (2013) Elaborations on a theory of human problem solving. Poster collection from the second annual conference on advances in cognitive systems. Baltimore, MD, pp 111–122