Stanford Heuristic Programming Project
Report No. HPP-83-20

March 1983

The Utility of Meta-level Effort

Jeffrey S. Rosenschein
Vineet Singh

COMPUTER SCIENCE DEPARTMENT
Stanford University
Stanford, California 94305

The Utility of Meta-level Effort

Jeffrey S. Rosenschein and Vineet Singh

Heuristic Programming Project
Computer Science Department
Stanford University
Stanford, CA 94305

Abstract

Meta-level control, in an Artificial Intelligence system, can provide increased capabilities and improved performance. This improvement, however, is achieved at the cost of the meta-level effort itself. To ensure an overall increase in system efficiency, the savings brought about at the base level cannot be exceeded by the effort at the meta-level. This paper outlines a formalization of the costs involved in choosing between independent problem-solving methods; the cost of meta-level control is explicitly included. It is shown that when meta-level effort is related to its efficacy, there exists an amount of this effort that should optimally be expended. Too much or too little meta-level effort can result in a loss of overall system performance.

Introduction

It is often desirable for Artificial Intelligence systems to make use of explicit knowledge about what they know; this *meta-level knowledge* allows a program to direct its own activities in an informed and efficient manner [Davis 76, Barr 82]. The use of meta-level knowledge by a system to control its own actions is called *meta-level control*.

While meta-level control can clearly be beneficial to the overall performance of a system (by both capability and performance criteria), it is not achieved without cost. Every system resource that is expended at the meta-level adds to the overall cost of a problem's solution. If we are to gain efficiency through the use of meta-level effort, we must be sure that what is saved at the base level is not canceled by what is expended at the meta-level.

This paper presents a straightforward model of decision-making, then develops a formalization of cost that explicitly includes the cost of meta-level effort. A more generalized formalization of meta-control (that includes the probability of making the correct choice) is then examined; criteria are given for when, and to what extent, this type of control effort should be expended. Finally, several examples demonstrate that too much or too little meta-level effort decreases overall efficiency; included among these examples is an analysis of task allocation strategies in distributed systems.

Ordering Independent Methods

The Simple Model

Assume that a computer system has at its disposal two independent methods for solving a problem; neither method is guaranteed to succeed, but probabilities of success exist for each one. Let us call the two methods x and y; let p_x be the probability that x will satisfy the goal, and let e_x be the expected cost of performing x (regardless of whether or not it succeeds). Define p_y and e_y in the same way for method y. Note that cost can be a function of any parameters associated with a method, such as time and storage. Barnett [Barnett 82] and Simon [Simon 75] have shown that the expected cost of executing x before y (i.e., executing x, and then, if necessary, y) is

$$E(xy) = e_x + (1 - p_x)e_y.$$

An analogous result holds for E(yx). The independence of the two methods ensures that e_x, e_y, p_x and p_y remain unchanged regardless of the order in which x and y are performed.

The decision to try x before y should be made when this ordering has a lower expected cost than the alternative, that is, when $E(xy) < E(yx)$; of course, either method can be tried first if $E(xy) = E(yx)$. If φ_z is defined as p_z/e_z (for $z = x$ and $z = y$), we find (again following Barnett) that x should be tried before y precisely when $\varphi_x > \varphi_y$, since

$$E(xy) < E(yx)$$
$$e_x + (1 - p_x)e_y < e_y + (1 - p_y)e_x$$
$$p_y/e_y < p_x/e_x$$
$$\varphi_y < \varphi_x.$$

The result generalizes to n independent methods: compute φ_i for $1 \leq i \leq n$, and try methods ordered by their φ's, largest φ first. All of the above results are developed in [Simon 75] and explicated in [Barnett 82]; Barnett also computes the savings of performing the methods in their optimal ordering versus performing them in their alternative ordering, namely:

$$S = |E(yx) - E(xy)| = |p_x e_y - p_y e_x|.$$

He shows that if the optimal ordering is used instead of a totally random choice between x and y, the expected savings is S/2. This, then, is the "value" in this simple model of control knowledge used to pick the ordering.

Including Control Cost Explicitly in the Simple Model

Let us consider the cost of the decision procedure explicitly in our savings computation. We define E(dxy) as the expected cost of performing a decision procedure d, followed by method x and (possibly) method y (i.e., if x fails). E(dxy) is thus equal to $e_d + e_x + (1 - p_x)e_y$, where e_d is the expected cost of the decision procedure. If the system were going to perform method y first, but because of the decision procedure discovered that method x was preferable as a first alternative, the savings would be equal to

$$\begin{aligned} S &= E(yx) - E(dxy) \\ &= [e_y + (1 - p_y)e_x] - [e_d + e_x + (1 - p_x)e_y] \\ &= p_xe_y - p_ye_x - e_d. \end{aligned}$$

Conversely, if x would have been performed first but the decision procedure tells us y is the correct choice, our savings $S = p_ye_x - p_xe_y - e_d$. Thus, the true expected savings S is $|p_xe_y - p_ye_x| - e_d$ (where $|q|$ is the absolute value of q).

In fact, this is not quite a realistic case; it can be assumed that, with no control knowledge expended, there is an equal chance of either method being tried first. Therefore, our savings is equal to $E(yx)/2 + E(xy)/2 - E(dxy)$ (or, conversely, the last term might be $E(dyx)$), and we have

$$S = (|p_xe_y - p_ye_x|/2) - e_d. \tag{1}$$

Extending the Model with Decision Probabilities

In general, we cannot assume that the decision procedure d finds the correct course of action (i.e., finds the true values of φ_x and φ_y). Assume that the decision procedure claims that $\varphi_x > \varphi_y$, but the likelihood that this is actually true is some probability p_d. In this case, our expected savings is

$$p_d(p_xe_y - p_ye_x) + (1 - p_d)(p_ye_x - p_xe_y) - e_d$$

i.e., 1) the expected gain $(p_xe_y - p_ye_x)$ when the decision was correct, times the probability that it was correct, plus 2) the expected gain ($p_ye_x - p_xe_y$, a negative quantity) when the decision was incorrect, times the probability that it was incorrect, minus 3) e_d, the decision cost. For the sake of convenience, let us define W to be $|p_xe_y - p_ye_x|$. Now, taking into account the fact that x and y may actually be reversed in the above equation (depending on the true sizes of φ_x and φ_y), and rearranging, we get the expected savings to be

$$S = (2p_d - 1)W - e_d,$$

and the savings over what we would get with a random choice is

$$S = [(2p_d - 1)W/2] - e_d.$$

Note that when $p_d = 1$, this result generalizes to Equation (1) above.

Probability of Correct Choice as a Function of Meta-level Effort

In realistic situations, the probability of having made the correct choice between methods, p_d, is some function of the amount of effort expended at the meta-level, i.e., $p_d = f(e_d)$. Placing this new function in our equation above, we find that our savings when using a variable amount of decision effort is

$$S = [(2f(e_d) - 1)W/2] - e_d. \qquad (2)$$

Since we want to maximize S as a function of e_d (i.e., get the maximum savings for our meta-level effort), we take the derivative of S with respect to e_d, set the result equal to zero, and find that, at the point where savings are maximized,

$$f'(e_d) = 1/W. \qquad (3)$$

Figure 1 provides a graphical representation of the potential relationship between decision effort and savings.

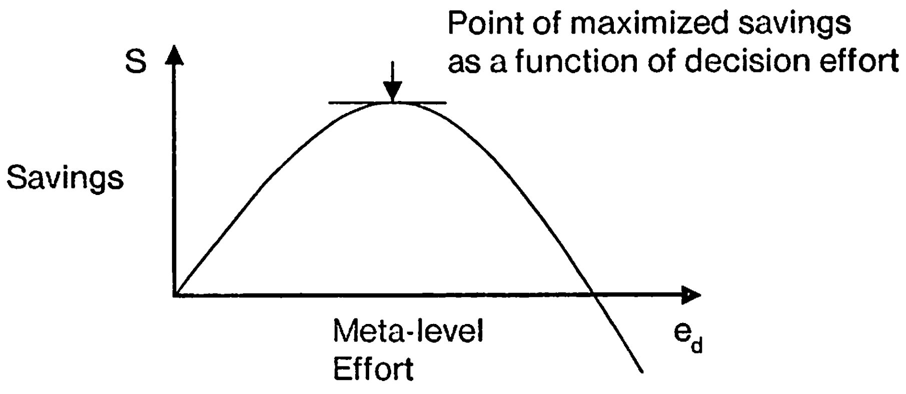

Figure 1: Savings as a function of meta-level effort

Note that there may be no value of e_d that satisfies Equation (3). This is because the equation is derived under the assumption that the derivative of S with respect to e_d can be set equal to zero; in fact, this is not always the case, as will be seen in the second example below.

Practical Implications

To precisely determine optimal meta-level effort using the above analysis, one has to know the value of W and the relationship between p_d and e_d. This is not necessarily a realistic assumption. In particular, the most likely way of knowing W is to know the values of p_x, e_x, p_y and e_y; in that case, a trivial decision procedure can be used to choose between the methods.

In practice, one would use estimates of W and the relationship between p_d and e_d to compute an approximate value of e_d at which optimal savings result. Heuristics may be employed to make these estimates. In addition, it may be possible for a system to refine estimates and develop an approximate model of p_d as a function of e_d through the collection of statistics.

In most realistic cases, there will not actually be a known relationship between p_d and e_d. Even then, the analysis outlined above is useful. If one has two or more points at which the savings brought about by meta-level effort can be computed, the analysis can be used to make the proper choice. This case will be highlighted in our task allocation example below.

An Example

Let us now use Equation (3) to analyze an example of meta-level control. It is often the case in reality that a small amount of meta-level effort results in substantially improved performance, while there are diminishing returns for subsequent effort. A reasonable form for $f(e_d)$ that captures this relationship is shown in Figure 2. In this case, the probability of making the correct decision is 0.5 when no decision effort is expended (this will be true if the choice is made randomly, and if x and y have equal probabilities of being the correct choice). Moreover, as more effort is expended at the meta-level, the probability of making the correct decision asymptotically approaches 1. A function that displays these characteristics is $f(e_d) = 1 - [e^{(-e_d)}/2]$, for e_d on the interval $[0,\infty)$. The derivative of the function is $e^{(-e_d)}/2$; plugging this value into Equation (3) and solving for e_d, we find that $e_d = \ln(W/2)$. This is the amount of control effort that produces maximum savings, given that p_d is the aforementioned function. The savings achieved using this amount of control effort is $[(W-2)/2] - \ln(W/2)$.

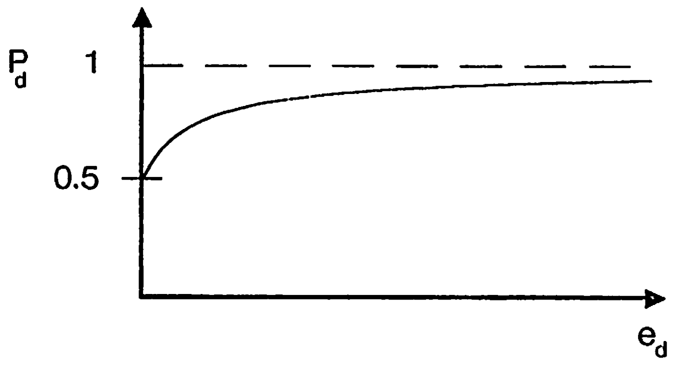

Figure 2: An example of the relationship between probability of correct decision and meta-level effort

Note that it is possible, for some W's, to use the above method and find that e_d should be negative to achieve maximum savings. Since, however, this falls outside the interval on which the function $f(e_d)$ is defined, we examine the interval end-point(s) to discover a maximum achievable savings. With the above function, for example, if e_d came out negative then maximum achievable savings would be with $e_d = 0$. In this case no meta-level effort would be expended, and there would be no savings at all.

A Second Example

In many realistic situations, a certain amount of meta-level effort will guarantee that the base level method with lower expected cost is chosen. Figure 3 illustrates a form for $f(e_d)$ that captures this; the linear relationship between p_d and e_d in the interval [0,v] is arbitrary, but certainly reasonable.

Formally, $f(e_d) = (0.5/v)e_d + 0.5$ for e_d on the interval [0,v], and $f(e_d) = 1$ on the interval (v,∞); v in this instance is the predetermined constant at which the meta-level effort guarantees the correctness of the base level choice. Since $f(e_d)$ is constant in the interval (v,∞), savings (S) as given by Equation (2) decrease as e_d increases. Therefore (and because there is no discontinuity for savings at $e_d = v$), the maximum savings (for any value of e_d) is obtained at the relative maximum in the interval [0,v]. Also, in the interval [0,v], S equals $(e_d/v)W/2 - e_d$, or equivalently $e_d([W/(2v)] - 1)$, using Equation (2) and $f(e_d)$ as defined here. S', the derivative of S, is therefore $[W/(2v)] - 1$.

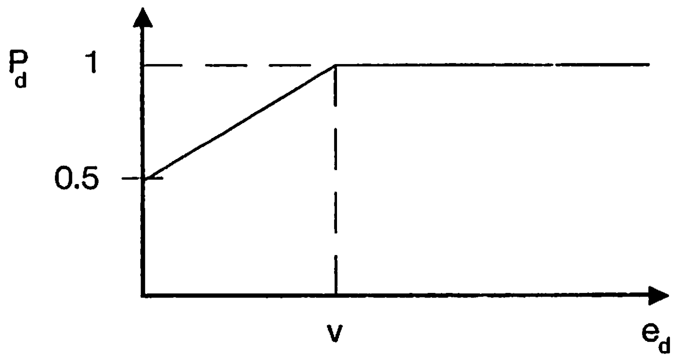

Figure 3: A second example of the relationship between probability of correct decision and meta-level effort

If $v > W/2$, then S' is negative; therefore, the relative maximum in the interval [0,v] lies at $e_d = 0$. Similarly, when $v < W/2$, S' is positive and the relative maximum lies at $e_d = v$. There is a particularly interesting situation when $v = W/2$. In this case, S' is zero in the entire interval and so is S. This means that any extra effort spent at the meta-level is exactly negated by an equivalent decrease in effort at the base level, and any cut in the effort at the meta-level is also exactly negated by an equivalent increase in effort at the base level.

Note that in this example it is not possible to simply set S' to 0 and solve for e_d, since e_d does not appear in S'. The relative maximum of e_d must be found at the edges of the intervals on which it is defined.

A More Realistic Example

Task Allocation in a Distributed System

One of the primary motivations of distributed systems is increased performance. Performance, in turn, is crucially dependent on the task allocation strategy used. Quite appropriately, much effort has been expended in the recent past to synthesize new task allocation strategies and to analyze them in terms of computational complexity theory [Lageweg 81, Mayr 81, Dolev 80]. We will focus on the following special case of the task allocation problem:

- There exists an arbitrary but finite number of identical processors.
- A finite set T of tasks need to be allocated to the processors. Associated with each task is the processing time required for the task. (The number of tasks is usually considered to be an order of magnitude greater than the number of processors.)
- There exists a precedence constraint over T defined by a partial order $\prec$. If t_i and t_j belong to T and $t_i \prec t_j$, then the execution of t_j cannot start until the execution of t_i is finished. (The precedence constraints mentioned here arise from data or control flow constraints.)
- Once a processor begins to execute a task, it works to completion unless the task execution is aborted for some reason. If a task is aborted, partial results cannot be used and the task must be processed from the beginning elsewhere.
- The optimality criterion is to minimize the time at which all tasks have been completed.

Consider at first that task descriptions are not available simultaneously, but rather that they flow into various processors from outside the system at possibly different times. One task allocation strategy is the following:

When a processor receives new task descriptions from outside the system, the descriptions are broadcast to the community of processors. Each processor maintains a list of task descriptions that it hears about along with any associated precedence constraints. A task is said to be available if it has no uncompleted predecessor tasks (defined by the precedence constraints), and if it is not already being worked on by some other processor. Whenever a processor is free, it looks at the set of available tasks and picks the one that heads the longest chain (in terms of the sum of processing times) of unexecuted tasks, broadcasts its intention of grabbing the task, and starts executing it. Any conflicts between processors are resolved by a preset ordering among them. Therefore, it is possible that a task executing at a processor may have to be aborted on receipt of a broadcast from another processor; let us assume that such multiple, partial executions of tasks do not create any problems. After completion of a task, the processor that completed the task broadcasts that it has done so.

The algorithm described above is more decentralized than an allocation strategy based, for example, on a contract net approach [Davis 81] and so potentially offers advantages of efficiency and reliablility over the latter. A detailed discussion of the comparative merits and demerits of the two approaches is, however, outside the scope of this paper.

The task distribution strategy mentioned here has been analyzed by Graham [Graham 69] for the case where all task descriptions are available simultaneously. Call this new algorithm A_1. If n is the number of processors, w_1 the completion time for the tasks using this algorithm A_1, and w_0 the minimum possible completion time provided by any algorithm, then Graham proves the following upper bound:

$$w_1/w_0 \leq 2 - (2/(n+1))\,;$$

i.e., the worst case for the completion time using A_1 is at most twice (asymptotically as the number of processors approaches infinity) the completion time using an optimal algorithm.

It is known that the problem of finding an optimal schedule for the case mentioned above is NP-complete [Lageweg 81, Garey 79]. The simplest exponential algorithm that finds an optimal schedule is perhaps the algorithm that enumerates all possible allocations and then picks the one that leads to the earliest completion time; call this algorithm A_{Exp}. The next subsection will compare this exponential algorithm with A_1 using criteria developed earlier in this paper for evaluating meta-level effort.

Choosing Among Distributed Task Allocation Algorithms

The total cost of any task distribution strategy may be thought of as consisting of two parts: *base level* cost, the "run-time" cost of actually performing the tasks, and *meta-level* cost, the "set-up" cost of deciding which task should be performed next (and by which processor). For example, the upper bound on the total cost of algorithm A_1 is given by

$$C_{A_1} < 2w_0 + O(s), \qquad (4)$$

where s is the number of tasks in the system. By comparison, the upper bound on the total cost of the exponential algorithm A_{Exp} is

$C_{A_{Exp}} \leq w_0 + O(k^s)$, where $k > 1$.

We would like to judge how much meta-level effort is warranted in the above case, that is, how much effort should be spent scheduling to maximize overall savings. Our particular formalization of costs is not directly applicable in this case, since increased meta-level effort affects base level costs (i.e., the execution of the tasks takes less time). In addition, our characterizations above are upper bound costs, not expected costs. Nevertheless, the overall theme of computing preferable meta-level effort, on the basis of total cost, remains.

Thus, although the second algorithm gives provably optimal performance at run-time, the meta-level cost may be prohibitively high. In particular, if the savings introduced by reducing run-time by (at most) w_0 are exceeded by the difference between exponential and linear costs of scheduling the tasks, the "non-optimal" strategy is preferable. C_{A_1} will be less than $C_{A_{Exp}}$ under a suitable combination of the following circumstances:

1. s, the number of tasks to be scheduled, is sufficiently large;

2. w_0, the optimal length of the run-time schedule, is sufficiently small.

We will choose to use algorithm A_1 when C_{A_1} is less than $C_{A_{Exp}}$, that is, when

$w_0 + c_1 s + c_2 < c_3 k^s + c_4$,

where c_1, c_2, c_3, and c_4 are the algorithms' relevant constants. For example, assume that $w_0 = 4$ hours, c_1 through c_4 are 1 (second) each, $k = 2$, and $s = 10$. In this case, the value of $w_0 + c_1 s + c_2$ greatly exceeds $c_3 k^s + c_4$, and the exponential algorithm will be chosen. Conversely, if w_0 were 1 minute and $s = 1000$, A_1 would be chosen.

There is another way of looking at the problem of deciding between task allocation strategies. The decision regarding how much meta-level effort to expend (which is actually a meta-meta-level decision) can be seen as an instance of our earlier analysis: call algorithm A_1 our method x, and the algorithm A_{Exp} our method y. In addition, consider both elements of task distribution cost to be at "base level" (e.g., $C_{A_1} = e_x$, and consists of the two terms in Equation (4)). Then the meta-level decision effort, e_d, is expended in trying to determine the relative costs of the two methods (since both methods are guaranteed to succeed, p_x and p_y are 1). The following are five ways to augment the decision-making process as it chooses between the two algorithms, each involving increased effort and returning an increased probability of making the correct choice:

1. Examine the number of tasks, s;

2. Compute the summation of these task times, which gives an upper bound on w_0;

3. Examine the nature of the precedence constraints over the tasks, since it may be a special case (e.g., no constraints, or a total ordering on all tasks);

4. Determine the exact constants on the costs of the algorithms (e.g., replace $O(s)$ by $57s + 7$);

5. Execute an entire algorithm, and get an exact value for the method's cost (this, of course, obviates the need for performing some of the base-level computation).

Each method involves an increased meta-level effort, and after completion increases the probability of making the correct decision. The exact amount of meta-level effort to expend will depend on the relationship between this effort, e_d, and the probability of success, p_d; as discussed above, this relationship could be approximated through empirical observation.

Conclusions

Meta-level control can be a valuable source of increased system efficiency, but only if exploited in the proper way. Too much or too little meta-level effort can result in a loss of overall system performance. To expend the optimal amount, we must analyze the relationship between total costs on the one hand, and improvement brought about by meta-level control on the other. In certain cases, it is possible to quantify this relationship and precisely determine the best amount of meta-level effort for the problem.

Acknowledgment

The original idea of examining the above issues arose in discussion with Mike Genesereth; in addition, the resulting conclusions have benefited greatly from his comments.

References

[Barnett 82]	Barnett, Jeffrey A. *How Much is Control Knowledge Worth?: A Primitive Example.* ISI Working Paper, USC/Information Sciences Institute, June, 1982.
[Barr 82]	Barr, Avron and Edward A. Feigenbaum (Eds.). *The Handbook of Artificial Intelligence.* William Kaufmann, Inc., 1982, chapter VII.
[Davis 76]	Davis, Randall. *Applications of Meta-level Knowledge to the Construction, Maintenance, and Use of Large Knowledge Bases.* PhD thesis, Stanford University, 1976. Memo AIM-283, AI Laboratory, and Rep. No. STAN-CS-76-552, Computer Science Dept., Stanford University; Reprinted in R. Davis and D. Lenat (Eds.), *Knowledge-based Systems in Artificial Intelligence*, McGraw-Hill, 1982, pp. 229-490.
[Davis 81]	Davis, Randall and Reid G. Smith. *Negotiation as a Metaphor for Distributed Problem Solving.* MIT A.I. Memo 624, Massachusetts Institute of Technology, May, 1981.
[Dolev 80]	Dolev, Danny. *Scheduling Wide Graphs.* Report No. STAN-CS-80-832, Department of Computer Science, Stanford University, December, 1980.
[Garey 79]	Garey, Michael R. and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness.* W. H. Freeman and Company, San Francisco, 1979.
[Graham 69]	Graham, R. L. Bounds on Multiprocessing Timing Anomalies. *SIAM Journal of Applied Mathematics* 17(2):416-429, March, 1969.
[Lageweg 81]	Lageweg, B. J., E. L. Lawler, J. K. Lenstra, and A. H. G. Rinnooy Kan. *Computer Aided Complexity Classification of Deterministic Scheduling Problems.* Technical Report BW 137/81, Mathematisch Centrum, 1981.
[Mayr 81]	Mayr, E. W. *Well Structured Parallel Programs Are Not Easier to Schedule.* Report No. STAN-CS-81-880, Stanford University, September, 1981.

[Simon 75] Simon, Herbert A., and Joseph B. Kadane.
Optimal Problem-Solving Search: All-or-None Solutions.
Artificial Intelligence 6:235-247, 1975.