Incorporating Resource Analyses into an Action System

Matthew Brand
Northwestern University
The Institute for the Learning Sciences
Evanston, Illinois
brand@ils.nwu.edu

Abstract

OOPS is a reactive planner which integrates sensory perception and action selection. A principal feature of the OOPS architecture is the use and discovery of cheap, diagnostic features to indicate opportunities, which are then verified in more expensive computations. This diagnostic relationship is established analytically and refined using tools from decision theory. The use and refinement of diagnostic features depends upon assumptions of conditional independence. However, in the case of a multiplanning agent (one which simultaneously pursues several goals), while conditional independence holds true for features vis-a-vis individual opportunities, plans may interact in the world, and conditional independence may not hold. In this paper we discuss how the knowledge needed to avoid detrimental action interactions can be incorporated into OOPS's inexpensive diagnostic computations, with benefits for robustness, performance, and learning.

Realtime action and multiplanning

The Optimizing Opportunistic Planning System (OOPS) is an action architecture designed to propel a purposeful agent through an environment that is rich with opportunities and threats. It is designed for realtime perception and multiplanning: at any given time, OOPS is juggling several ongoing plans, each in the service of a different active goal, and all selected by an integrated perception/decision system.

Multiplanning poses challenges to any planner, synthetic or reactive, that go beyond the normal difficulties of protecting satisfied subgoals. Interactions between concurrently executing plans can demolish any plan's hope for success. This is a particular concern for memory-based and reactive planners, which generally substitute pre-specified plans with protections already worked out for the machinery which ensures protections. Such planners are ill-equipped to deal with resource conflicts between unrelated plans. This is a particular concern in our research, since the design of OOPS emphasizes the production of behavior from perceptual rather than inferential mechanisms.

One of the central thrusts of the design of OOPS is finding and relying upon very cheap perceptual features that can trigger actions with high potential utility[1]. This reflects a research assumption that an agent may be successful in its world with only a limited ability to consider complex interactions between states and actions. In order for this to be true, there must exist a wide range of cheaply computed and conditionally independent features, which can account for the greater part of the agent's useful behaviors. In the following discussion, we discuss the strengths and apparent limitations of this hypothesis, and how to deal with the fact that multiplanning violates the conditional independence assumption, without departing significantly from our perception-intensive framework.

OOPS' view of the world

The world that OOPS is being designed for is not unlike the world of an animal: changing, perceptually rich and noisy, requiring constant monitoring and rapid action. As previous work in reactive planning and action architectures [Brooks, Connell, & Ning, 88, Maes & Brooks 90, Firby 89, Hammond Converse & Martin 90, Agre & Chapman 87] has shown, we have found that being tightly coupled with such a demanding environment shifts the emphasis in planning from correctness and completeness to concerns about ecological adequacy, flexibility, and reaction speed. Realtime planning means that quickly identifying usable and desirable actions is preferable to synthesizing and verifying plans, and that the commitment of resources to potential actions—including computational attention—must reflect those actions' applicability and utility. There is a premium on the ability to rapidly identify and execute, as well as suspend, resume, or abandon useful plans in response to the face of novel circumstances.

[1] For similar views on the significance of low-level features in representation, see [Forbus & Gentner 90]

Integral perception and action

The tight coupling between an action agent and its world should be reflected by a tight coupling between sensory and motor processing. OOPS meshes the ethological model of drive hierarchies [Tinbergen 51, Baerends 76, Dawkins 76] with connectionist-like perception networks [Rumelhart &McClelland 86], in order to achieve a smooth integration of perception and action selection. The architecture embodies the idea of *computational economy*: an expected value can be calculated for every prediction of a feature or action opportunity, and then used to determine which features and proposed actions should receive computational attention. This yields a uniform and principled procedure for allocating resources to promising perceptual and action hypotheses.

OOPS epistemics: diagnostic features

The OOPS network includes a hierarchy of "quick and dirty" diagnostic features and more expensive, precise features. A cheap feature, when signalled, can be validated by a more expensive feature [Birnbaum 86, Sussman 75]. Features can be more expensive in their CPU requirements, the larger array of subfeatures they depend on, or in that they take longer to detect in the world and thus have less predictive value. In OOPS the main thrust of learning is to validate existing feature relationships, and to find ever cheaper features that detect situations of utility to the agent. As this proceeds, attention focusing and its motor equivalent, action selection, become increasingly efficient and accurate.

Action selection is a two-tiered process. First, values are obtained for all features and drives[2] then used to identify the most highly applicable and desired actions. This computation is very cheap; the update cycle is constrained to yield a result in time linear in the size of the OOPS network. In the second phase, proposed actions are verified according to more elaborate specifications of conditions of applicability, and then effectors are allocated to selected actions. The specifics of attention focusing, representation, and information flow are discussed in [Brand & Birnbaum 90].

Conditional independence The use of cheap diagnostic features to rapidly winnow the range of action down to a few promising choices is a response to the need for a agent to be "on the ball," able to act fast on new information. In order for diagnostic features to be worth their costs, assumptions of conditional independence must hold: features must correctly predict opportunities in the *vast majority* of common situations, without requiring heaving computation. This means that the feature should be computable with little regard to context, yet still be applicable in many contexts. For example, innate releasing mechanisms in animals such as the red breast of the courting frigate bird are good conditionally independent features, because they are easily detected, yet have the same meaning in almost all contexts [Eibl-Eibesfeldt 75]. As will be discussed below, assumptions of conditional independence are even more important to learning.

Conditional independence and learning For learning, a third phase follows in which the results of the two phases are compared, and the resultant information is incorporated into estimates of the predictive value of features using signal-detection theory [Green & Swets 66]. The diagnosis of a feature with regard to any other feature or action is classified into True Hit, False Alarm, Correct Rejection, or Missed Opportunity, and this is used to derive the expected value of the feature [Brand 91]. This information is then incorporated into connection strengths between them.

Resource conflicts The rate and success of learning is partly determined by the degree that features actually are conditionally independent. A problem in the operation of the OOPS simulator is that though its first pass correctly chooses a set of applicable actions, many of these actions fail to execute after inspection because of conflicts over effectors and external resources. That is, with regard to resource management, these actions are *not* conditionally independent. At first blush, this merely means a waste of computation; it would be preferable if the first pass proposed fewer actions with likely conflicts. However, the problem is more serious in that this generates unwanted false alarms: Proposed actions fail even though their conditions of applicability are satisfied. Unwanted false alarms retard learning the relationships of predictive features, because the learning algorithm construes false alarms as incorrect diagnoses of opportunities, *independent* of other plans that may be executing. Such failures cannot simply be discounted or re-interpreted as semi-successes. Discounting means losing information important for learning. Re-interpreting means trying to represent in a connection a relation that does not hold between the feature and action it connects: that under unknown conditions some other proposed action may usurp effectors needed to execute the action.

Multiplanning

Inside the agent, the primary challenge of multiplanning is the selection, execution, and retirement of plans. In OOPS this is achieved through a simple extension of the attention focusing system, which enforces an economy of internal resources, notably

[2] In actuality, one of the principal advantages of the notion of *computational economy* is that only selected features are computed, according to their expected utility and the availability of CPU resources. Nonetheless, values are *available* for all features, and are accurate in proportion to the agent's interest in each feature.

computation and memory.

Outside the agent, the primary challenges arise from the need to coordinate actions: the planner must avoid conflicts over resources. Depending on the number of effectors an agent has under independent motor control, it may be pursuing several plans simultaneously, interweaving plans, or using different plans to control separate effectors in the service of the same goal.

Where there is a concurrency of action, interactions between plans under execution become a serious concern. One may walk and chew gum successfully, but whistling and chewing gum is an uncertain proposition. This is a simple conflict over an effector. More subtle conflicts arise over resources frequently manipulated by separate effectors: A musician can certainly sing and play at the same time, but doing vocals over a solo is a bad idea because one may drown out the other, or both may be too complicated to listen to simultaneously. There is limited space in the dynamical and informational bandwidth of a listener's hearing. This must be learned by musicians just as toddlers learn that banging pots obscures the content of their screams for attention.

Interactions are an equally difficult problem for classical planners. Consider a case-based classical planner which must pursue multiple plans. Using a memory store of useful plans instead of synthesizing new ones becomes less and less advantageous, since the overhead of checking interactions between known plans is just as bad as the cost of doing such checking when synthesizing new plans. Nor will simple solutions help. As pointed out above, it is not satisfactory to simply block actions that require already allocated resources. It is entirely reasonable to interrupt a walk to kick a soccer ball back to the game it rolled out of, but somewhat less reasonable to do so when jogging, and not very wise to do so when being chased. The process of resource allocation must account for varying priorities.

The fact that plans have different resource usages over their execution course further complicates the matter: One cannot lock up every effector that an action will require over its execution life right from the outset. Some actions have very long execution courses, and will run concurrently with, or be punctuated by, more staccato actions. The effectors a plan uses near the end of execution shouldn't be reserved from the beginning. For example, running deep to catch a football should not preclude using one's hands to repel opposing players on the way. Nor should resources be completely available to be usurped by an unrelated plan: while in the workshop gluing wood, one does not want to "notice" an opportunity to fix a table using nearby clamps, only to find that none are left to take the place of one's hand holding the wood together.

Recognizing Resource Conflicts

The scope of this problem has had a substantial impact on our assumptions about what makes for an adequate epistemology of the dynamics of an agent's world. It is prohibitively expensive to inferentially predict the consequences of action interactions, yet there is a strong motivation to incorporate some sort of knowledge about conditional dependence into the our planner.

Noting that the greater range of human activity does not require provably correct planning—people routinely make planning errors and action slips[3] [Norman 88] and survive—one plausible hypothesis is that much of the knowledge about conditional dependencies is brought to bear on planning in a manner more associative, and less accurate, than truth-preserving inferential reasoning. That is, up to a point, a planner benefits from from knowledge about how its actions lock up and consume resources, but that point does not extend as far as costly inferential reasoning in all cases. Thus we have focused on moving such knowledge out of the expensive action-validation process and pushing it down into the computation-efficient process of action-proposal.

Looking to ethology

> *Animals avoid the potential problem of ambivalence; stimuli appropriate for more than one behavior may be present but the animal usually can only perform one behavior at a time.*[Toates 80]

What makes an animal so good at picking a workable behavior for each situation? The knowledge-poor answer is that things are stacked so that all possibilities can be considered in parallel. The knowledge-rich answer is that things are arranged so that the most appropriate possibilities are considered first. Cognitive scientists might attribute this to *priming*, AI researchers might say *indexing*. Ethologists tend to talk about *suppression*: Given the kind of situation the animal is in, a whole range of behaviors will be suppressed. For simple animals, there are perhaps 10-50 kinds of situations: feeding, predation, fight, flight, courtship, mating, gestation, sleep, birthing, patrolling, etc. Each of these has characteristic resource needs, and behaviors with inimical resource-consumption patterns are suppressed.

Avoiding resource conflicts

Consider a very simple example from the low end of the food chain: The gastropod *Pleurobranchaea* is a voracious eater. It will routinely abandon mating if food appears nearby. In its evolutionary history, energy intake probably was its greatest challenge. When laying eggs, however, it must suppress feeding

[3] Planning errors and action slips correspond to failures to preserve external and internal state, respectively.

to protect its own progeny. In its very simple world, the short-term resource strategy of energy intake (eat everything within range) must be subordinated to the long-term resource strategy of propagating its genes (stabilize the environment until the eggs hatch).

To take another example, mobility is a resource that many animals will temporarily sacrifice for extra manipulative capacity (e.g. woodpeckers wedge nuts into corners so that they can peck them, beavers wedge themselves against branches to gain leverage in fitting together new dam parts). When frightened, woodpeckers will avoid places in their environment that have been useful for wedging, but they will go near other places that are significant for them. Presumably, fear-related behaviors suppress behaviors that relate to reduction of mobility.

The key observation here is that suppression inhibits behaviors with incompatible resource usage patterns. Usually, these rival behaviors fail to become sufficiently active to claim effectors. Mapping this onto the two-tiered operation of OOPS, it is likely that suppression operates in the cheaper computations which *find* applicable actions, rather than the more expensive *validation* of proposed actions. In this view, suppression operates in the realm of memory more than it does in realm of reasoning. Although this provides only a weak protection from detrimental interactions, it may suffice to account for enough successes to make an agent environmentally viable.

Finally, in business planning, time-sensitive behaviors should suppress the consideration of behaviors that treat time as a limitless or free resource. An example lies in the reaction of liquor retailers to the alcohol sales tax imposed at the beginning of 1991. An owner of a large chain of liquor stores in Milwaukee was interviewed a few months prior to the date. He was stocking up in September for a sale in December. The reporter asked him why not have a long, huge sale to beat the deadline. The retailer answered that sales take an unpredictable amount of time to unload inventory; it was better to rent a warehouse, stock it up with pre-tax beer, and sell it off when consumer awareness of the tax was at its highest. In this case, a sale with specific time costs is preferable to one with indeterminate time costs, because the latter may not satisfy its goal before the deadline. The latter plan will also fail because it ties up many other resources, including store space, for its duration. Although the retailer's explanation may refer to more sophisticated reasoning about resource tradeoffs, suppression of inimical resource-strategies is *sufficient to produce the preference.*

Suppressing resource competitors

Most behaviors can be characterized in terms of the resources that they conserve and the resources that they expend. For example, hoarding conserves the potential energy of food at the expense of the time spent scavenging, retrieving, and hiding. Once these characterizations are available, mutually inhibitory relations should hold between behaviors with incompatible resource characterizations. This will reduce the likelihood of conflict, and thus failed actions.

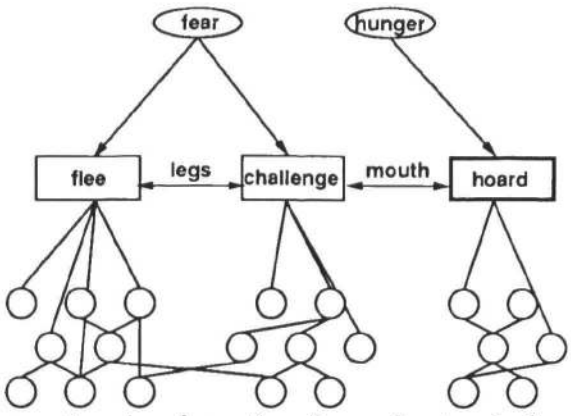

Figure 1: A schematic of a subnetwork from an OOPS simulation of an animal that must keep up both its territory and its food supply. The circles represent feature nodes; flee, challenge, and hoard are action nodes; mouth and legs represent effectors. Fear and hunger are drive nodes that boost the chances of their servant actions threshholding.

Resource characterizations can in many cases be deduced from the effectors that are involved. In the current set of OOPS simulations, where the domain is animal behavior, all the needed resource characterizations thus far have been available this way. Each effector is good for manipulating a particular kind of resource. A plan's resource consumption can be derived from the spans of time that it monopolizes various effectors. For example, consider the plan carry-back-to-nest. This monopolizes the jaw for the duration of the trip, and thus blocks access to, and use of, resources associated with the mouth: dexterity, weaponry, food. In simple animals, where all effectors are just bodily parts, one can produce a resource analysis purely by examining all the behaviors that compete for the same effectors. In many human domains, there are resources that are not explicitly manipulated by effectors, thus the analyses are more complicated.

Consider a dog carrying a tasty bone. Another dog approaches. Normally the first dog would approach to inspect and challenge, out of territoriality. In this case, it makes more sense to run away with the bone. Why? What is it that causes the flee action to threshold before the challenge action? We can take figure 1 to be a simplified model of the drives and behaviors involved. Normally, when fear is active and a foreign dog is perceived, challenge would threshhold higher than flee. It would be validated first and gains control of the legs. Flee would also be validated, but

then fail to claim the **legs**, and thus signal a false alarm. But in this case, **challenge** is being suppressed by **hoard**, because the two actions tend to have resource conflicts. It fails to threshhold. **Flee** threshholds and the the system chooses to run, without the additional overhead of considering **challenge**. In this example, the conditionality expressed in the lateral inhibition between actions suffices to bias the network in favor of flight.

Generalized resource analyses

In animals, effectors are synonymous with body parts, and it is easy to propagate information about the full catalog of effectors back into the network, where it takes the form of suppression links between actions. In modelling more abstract human tasks, the use of tools expands the range of potential effectors. In these cases, the network must be updated as new tools become available. However, this is probably a low overhead operation, compared to savings in computation gained from such additional knowledge.

---

for each behavior B
  for each effector E used by behavior B
    for each resource R consumed by effector E
      for each effector E' indexed under resource R
        for each behavior B' which uses effector E'
          if R is a { *monopolizable*, *exhaustable*, *core*, *time* } resource
            increase inhibition between B and B'
          if R is a { *replenishable*, *self-replenishing*, *sharable*, *renewable*, } resource
            decrease inhibition between B and B'

Figure 2: A simple heuristic for establishing a priori resource conflicts when constructing an action network. This algorithm depends on an abstract characterization of resource depletion and replenishment. *Renewable*, *replenishable*, and *self-replenishing* refer to resources that can be restored instantaneously, willfully, and automatically, respectively. An example of each is river water, food stores, and food crops in the management of a town. A preliminary typology of resources appears in [Brand 90].

For a model of an agent in the financial industry, effectors are what economists call financial "instruments." Options, contracts, holds, voting blocks, etc., are all instruments. An interesting aspect of instruments is that they are largely understood by those who use them as devices for exchanging resources. In any newspaper discussion of devices and their use, they are discussed almost exclusively in terms of the resource trade-offs that they embody. Options preserve opportunities at the loss of cash or credit. Voting blocks are nonliquid assets that reduce opponents' control of other companies.

Consider two corporations in a takeover battle, predator and prey. Two of the many plans for fighting a takeover are (1) raise one's own stock price through buy-backs, and (2) achieve a voting bloc of the predator's stock, and lead a shareholder vote to challenge the takeover internally. The instruments used here are voting blocks and options, particularly options on one's own stock and on that of potential predators.

Both of these instruments consume liquidity. In fact, to be effective, both devices consume indeterminate amounts of liquidity, since it is not clear in advance how much one's own share price must be driven up to drive off a suitor, nor how many voting shares are necessary to lead a shareholder revolt against the suitor's takeover plans. Even if sufficient liquidity is available for exercising both options at the outset, since both plans may require more cash or credit, it would be foolish to pursue both tracks.

An analysis of either instrument shows that they both require large amounts of liquidity. This information can be propagated back to both plans, where the *lack* of a specification of how much liquidity is needed shows that both plans require indeterminate amounts of money. Thus labelled, an inhibitory connection is added so that in the future, they are less likely to undermine each other through concurrent execution. Note that this is not a conflict over instruments, but the underlying resources that they consume.

Although both plans consume a resource of general universality—financial liquidity—the same treatment applies to more specific resources. Alternative plans for taking legal action should have the same mutually inhibitory relationship because of possible conflicts over the resources of the legal department, money, and intra-office information flow.

Summary

OOPS is an action architecture designed with an emphasis on the perception of opportunities. It reflects the belief that inferential mechanisms are not efficient enough to produce realtime behavior over a course of situations routinely encountered in a robot's world. The system attempts where ever possible to substitute inexpensive computations for expensive computations, without loss of utility. Cheap diagnostic features yield probablistic indications of opportunities, and then more expensive mechanisms are invoked to verify the opportunities and select among proposed actions. Decision-theoretic methods are used to learn the probablistic value of each feature's diagnosis; the results are used to allocate computation to hypotheses about opportunities in the world. Both the utility and the learnability of these features depends upon assumptions of conditional independence. Conditional relations between features are only handled after the winning hypotheses have thresholded. However, the conditional independence of hypotheses does not nec-

essarily hold; frequently, viable plans are considered that have conflicting requirements. They may need the same effectors, or need different effectors that usurp the same resources. Anticipating and handling such conflicts is far to expensive to do inferentially. Rather than leave it up to expensive action validation to protect against such conflicts, it is much more economical to "push down" this knowledge into the cheap pre-thresholding computation, where inhibitory connections between action hypotheses bias perception against seeing conflicting opportunities. This insulates the multiplanner against the twin problems of ambivalence and overextension.

Acknowledgments

The ideas presented here were developed in collaboration with Larry Birnbaum. Thanks to Bruce Krulwich, Gregg Collins, Louise Pryor, and Michael Freed for many useful discussions. This work was supported in part by the Defense Advanced Research Projects Agency, monitored by the Air Force Office of Scientific Research under contract F49620-88-C-0058. The author is supported by a National Science Foundation Graduate Fellowship. The Institute for the Learning Sciences was established in 1989 with the support of Andersen Consulting, part of The Arthur Andersen Worldwide Organization. The Institute receives additional support from Ameritech, an Institute Partner, and from IBM.

References

[Agre & Chapman 87] Philip E. Agre and David Chapman. Pengi: An Implementation of a Theory of Activity. In *Proceedings of AAAI-87*, 1987.

[Baerends 76] G. P. Baerends. The functional organization of behavior. *Animal Behavior.* **24**:726-735. 1976.

[Birnbaum 86] Lawrence Birnbaum. *Integrated Processing in Planning and Understanding.* PhD Thesis, Yale University Computer Science Department, 1986. Available as Report RR-489.

[Brand 90] Matthew Brand. A Typology of Planning. In R. Schank, *et. al.*, A Content Theory of Memory Indexing. Institute for the Learning Sciences Tech Report #2. 1990.

[Brand 91] Matthew Brand. Decision-Theoretic Learning in an Action System. To appear in the *Proceedings of the 1991 Machine Learning Workshop.*

[Brand & Birnbaum 90] Matthew Brand and Lawrence Birnbaum. Noticing opportunities in a rich environment. In *Proceedings of the 12th Annual Conference of The Cognitive Science Society*, 1990.

[Brooks, Connell, & Ning, 88] Rodney Brooks, Jonathon Connell, and Herbert Peter Ning. A second generation mobile robot, AI Memo 1016, MIT Artificial Intelligence Laboratory, 1988.

[Eibl-Eibesfeldt 75] Irenaus Eibl-Eibesfeldt. *Ethology: The Biology of Behavior.* New York: Holt, Rinehart and Winston, Inc., 1975.

[Dawkins 76] Richard Dawkins. Hierarchical organization: a candidate principle for ethology. In P.P.G. Bateson and R.A. Hinde *(eds.) Growing Points in Ethology*, pp. 7-54. Cambridge University Press, 1976.

[Firby 89] R. James Firby. *Adaptive Execution in Complex Dynamic Worlds.* PhD Thesis, Yale University Computer Science Department, 1989. Available as Report RR-672.

[Forbus & Gentner 90] Kenneth Forbus and Dedre Gentner. Structural Evaluation of Analogies: What Counts? In *Proceedings of the 11th Annual Conference of The Cognitive Science Society*, 1989.

[Green & Swets 66] David M. Green and John A. Swets. *Signal detection theory and psychophysics.* New York: J. Wiley & Sons. 1966.

[Hammond Converse & Martin 90] Kristian Hammond, Tim Converse, and Charles Martin. Integrating planning and acting in a case-based framework. *Proceedings of the Eighth National Conference on Artificial Intelligence,* Boston, MA, pp. 292-297. 1990.

[Maes & Brooks 90] Pattie Maes. Learning to Coordinate Behaviors. In *Proceedings, AAAI-89.* 1990.

[Norman 88] Donald A. Norman. *The Psychology of Everday Things*, New York: Basic Books, Inc., 1988.

[Rumelhart &McClelland 86] David Rumelhart and James McClelland *(eds.). Parallel Distributed Processing.* Cambridge, MA: Bradford Books. 1986.

[Sussman 75] Gerald J. Sussman. *A Computer Model of Skill Acquisition.* American Elsevier, New York, 1975.

[Tinbergen 51] Niko Tinbergen. *The Study of Instinct.* Oxford University Press. Oxford. 1951.

[Toates 80] Frederick M. Toates. *Animal Behavior—A Systems Approach* New York: John Wiley & Sons, 1980.