Tagging in Metaheuristics

Ben Kovitz
Fluid Analogies Research Group
Indiana University
512 North Fess Avenue
Bloomington, Indiana 47408 USA
bkovitz@indiana.edu

Jerry Swan
Computing Science and Mathematics
University of Stirling
FK9 4LA Scotland UK
jerry.swan@cs.stir.ac.uk

ABSTRACT

Could decisions made during some search iterations use information discovered by other search iterations? Then store that information in *tags:* data that persist between search iterations.

Categories and Subject Descriptors

I.2.8 [**Problem Solving, Control Methods, and Search**]: [Heuristic methods]; G.1.6 [**Optimization**]: [Stochastic programming]

General Terms

Algorithms, Design

Keywords

metaheuristics; tags; design patterns

1. TAGGING

1.1 Problem

When early decisions in a sequence affect what is possible in later decisions, the early decisions can usually be made better if informed by the results of later decisions.

Benefit of hindsight occurs in two ways in metaheuristics: when constructing candidate solutions that are themselves sequences of decisions, such as paths or graphs (constructive search), and when choosing which of a set of variations of candidate solutions are most likely to be fruitful (perturbative search).

An example in a constructive search: When attempting to construct a short path through a graph, each decision is the choice made, at one vertex, of which edge to follow to reach the next vertex. A poor decision at an early vertex can limit all decisions made at later vertices to constructing unacceptably long paths. But there's no way to know this before exploring them.

GECCO'14, July 12–16, 2014, Vancouver, BC, Canada.

ACM 978-1-4503-2881-4/14/07 ...$15.00.
http://dx.doi.org/10.1145/2598394.2609844.

An example in a perturbative search: A genetic algorithm searching a high-dimensional space for an $\vec{x}$ that produces a minimal $f(\vec{x})$ can produce many more variations (mutations and crossovers) in each generation than it can feasibly explore in that generation. Some lead to fruitful regions of the fitness landscape and others don't. There's no informed way to choose which variations are most worth exploring without actually exploring them.

In iterative searches, previous explorations of sequences of decisions have often passed through the same decision points and already discovered results of some of the available choices.

A real-life illustration: While searching for a path out of a cave, you might come to the same branching point many times without realizing you'd been there before. To keep yourself from re-exploring a branch that leads only to a dead end, you could write "dead end" on the wall the first time you backed out of it. The wall at the start of that branch is the most convenient place for this information, since that's the exact place where you need it.

1.2 Solution

During or immediately after exploring each decision sequence, record just-discovered facts which can benefit later searches. Store them in a way that makes them easy to look up when they are most likely to be relevant, such as when a later search reaches a decision point that was crossed during the current search.

Such records are called "tags", "pheromones", "annotations", or "descriptors". Tags form a secondary level of description beyond that of the candidate solutions themselves, suggesting which directions of search are most promising.

The simplest kinds of tags are markers that a solution has already been explored, as in tabu search (section 2.1), or a single number such as the average fitness of previous paths that passed through a given edge, as in ant-colony optimization (section 2.2).

There is, however, no upper limit to the complexity of the information that can be stored in a tag. Tags can hold any kind of information about results discovered so far, relevant facts from the problem domain, or any kind of estimate of the promisingness of a line of exploration. For example, a classification algorithm might tag several items as a single group, reflecting the hypothesis that the differences between them are unimportant and should be left out of the classification. Upcoming searches can build upon that hypothesis, perhaps adding further tags linking it to other hypotheses.

Eventually that hypothesis may be rejected and then the tag will be removed or itself tagged as a "dead end".

1.3 Consequences

1. The fitness landscape becomes progressively more hill-shaped (less "rugged") as more searches occur.

Each tag functions as a sign pointing the way toward more-fruitful searches or away from less-fruitful searches. Future searches skip over alternatives that appear good on the basis of information local to a decision point but that information from elsewhere suggests will lead to poor results. Thus the tags provide a communication channel from decisions that come later on a path to decisions that come earlier. Each tag expands the "local" knowledge available at each decision point to include some "global" knowledge of the region of the fitness landscape that each choice leads to.

2. An explicit representation of promisingness of different lines of exploration grows and improves, separately from evaluation of solutions.

In effect, the tags co-evolve with the candidate solutions. Tags direct and prioritize future searches, and future searches find information that helps make and refine the tags.

3. A certain proportion of computational capacity will be devoted to the overhead of maintaining, accessing, and interpreting the tags.

Since tag-related computation occurs on each iteration of a search, it slows down the algorithm's main inner loop. As long as recording and interpreting tags is cheaper than rediscovering the information in them, tags are likely a worthwhile addition to most metaheuristics.

4. Tags can encode domain knowledge.

One of the defining properties of metaheuristics is their problem-domain independence. However, it is well-known that domain knowledge is the key to ensuring the tractability of precisely the kinds of problems we wish to address with metaheuristics.

A set of tags provides a way for a search to capitalize on domain knowledge when it's determined to be relevant. For example, a metaheuristic for laying out components on a printed-circuit board can exploit a tag that certain pins are ground pins, and favor exploring layouts that orient the components so their ground pins are all on the same side.

2. EXAMPLES

2.1 Tabu search

Tabu search [4] maintains a memory of candidate solutions that have been recently evaluated, declaring a "tabu" against evaluating them again until sufficient new candidates have been explored. This keeps the exploration of variations in a perturbative search from repeatedly rediscovering the same local minima.

Some forms of tabu search tag specific elements of candidate solutions that have recently been varied, declaring a tabu against varying them again too soon ("recency-based memory"). This temporarily holds the search within or outside a neighborhood of solutions.

Other forms of tabu search ("frequency-based memory") track how many times a certain solution element has been present in explored solutions or how many times an element has been changed, swapped, deleted, etc. These tags last many search iterations, and can identify features of the fitness landscape such as that a certain element has a large or a small effect on fitness. This information suggests important aspects of the structure of the fitness landscape, such as where are the paths between regions with very different properties. This helps the algorithm choose between intensification and diversification strategies and tune them appropriately.

An illustration of the injection of explicit domain knowledge to a tabu-search algorithm for finding maximal subgraph isomorphisms between two finite-state automata is given in [9]. Since a node representing an accepting state can't map to a node representing a non-accepting state, any such mapping is declared permanently tabu.

2.2 Ant-colony optimization

Ant-colony optimization (ACO) [2] searches for the best paths through a graph by tagging each edge with a "pheromone": a number that correlates with the overall quality of recently searched paths that contained that edge. While constructing a new path, the algorithm favors edges with high pheromone numbers. Thus when choosing which parts of a path to vary, the algorithm tends to vary edges from bad paths more and edges from good paths less. The pheromones decay in order to prevent the search from converging too quickly on a suboptimal path.

ACO has a particular advantage in problems where the properties of the graph change while the algorithm is running. The pheromones change right along with the introduction, removal, or re-weighting of edges, directing the search toward the paths that are most favorable right now. This illustrates the advantage of maintaining a description separate from that of the candidate solutions and their fitness. The pheromones redirect the gradient of the search by amplifying information about which search *directions* are most fruitful. The fitnesses of candidate solutions alone don't contain this information.

A simple way to add domain knowledge to an ACO algorithm is to reduce the pheromone level for known bad elements of a solution. For example, in the Euclidean Traveling Salesman Problem, a path with a pair of crossed edges is known to be longer than the same path with those edges uncrossed [1, p. 1-12]. Simply reducing the pheromone level on edges that recently crossed speeds up the basic algorithm by several hundred times on standard large problems [10].

2.3 Model-driven searches

Another kind of tag consists of an explicit model of the problem domain or the fitness landscape, updated as candidate solutions are inspected and evaluated. For example, estimation-of-distribution algorithms [7] maintain an approximate map of the fitness landscape. This map takes the form of a probability distribution over the set of all possible solutions, where higher probability densities indicate regions where fitnesses tend to run high. Each time a candidate solution is evaluated, the algorithm adjusts the probability distribution to more closely match the fitnesses actually found. Instead of choosing new candidate solutions by varying old ones, these algorithms simply choose by sampling the current probability distribution.

3. OTHER KINDS OF TAGS

This pattern does not explain all kinds of tags.

A kind of tag that occurs in genetic programming serves as a label for a code fragment [8]. This enables another part

of the program to refer to it—say, to call it as a subroutine. This, in turn, facilitates the evolution of a modular architecture, which in turn facilitates evolvability. Without tagged code fragments, reference from one part of a program to another has to rely on mechanisms that are much more easily broken by mutation and crossover. For example, referring a certain number of instructions ahead or backward, or to an absolute location in a program, will likely produce a completely useless program whenever an instruction is inserted or deleted.

Reference tags are made even more robust by being inexact. When a program branches to an inexact tag, the interpreter simply finds the best match. This enables tagged modules and code that refers to them to evolve fairly independently without breaking each other. Almost-duplicate modules can coexist in the same program and eventually specialize for different purposes. Eventually they can get distinct tags that other parts of the program can evolve to take advantage of, but a reference won't break completely if it doesn't specify exactly the right tag.

A dizzying variety of tags are found in complex adaptive systems [5]. Cell-adhesion molecules dictate when a cell adheres to other cells and what other kinds of cells it adheres to; they play a fundamental role in growth and in the immune system [3]. Every colony of the African weaver ant has a distinct scent, which enables ants to tell if they're in their own colony or a foreign one [6]. Company logos enable customers to easily recognize businesses, and uniforms help them tell who are the employees and who are the customers.

Like the tags described in the first two sections of this paper, reference tags, group tags, timing tags, etc. are "administrative overhead" that guides the activity of the system. These tags are different in that they primarily serve the purpose of coordination between different parts of a candidate solution or between different agents. They don't primarily record knowledge gained in previous search iterations to save the system from having to rediscover it.

4. REFERENCES

[1] Applegate, D., Bixby, R., Chvátal, V., and Cook, W. *The Traveling Salesman Problem: A Computational Study*. Princeton Series in Applied Mathematics. Princeton University Press, 2011.

[2] Dorigo, M., and Stützle, T. *Ant Colony Optimization*. Bradford, 2004.

[3] Edelman, G. *Topobiology: An Introduction to Molecular Embryology*. Basic Books, 1993.

[4] Glover, F., and Laguna, M. *Tabu Search*. Springer US, 1997.

[5] Holland, J. H. *Hidden order: How adaptation builds complexity*. Basic Books, 1995.

[6] Hölldobler, B., and Wilson, E. O. Colony-specific territorial pheromone in the african weaver ant Oecophylla longinoda (Latreille). *Proceedings of the National Academy of Sciences 74*, 5 (1977), 2072–2075.

[7] Larrañaga, P. A review on estimation of distribution algorithms. In *Estimation of distribution algorithms*. Springer, 2002, pp. 57–100.

[8] Spector, L., Martin, B., Harrington, K., and Helmuth, T. Tag-based modules in genetic programming. In *Proceedings of the 13th annual conference on Genetic and evolutionary computation* (2011), ACM, pp. 1419–1426.

[9] Swan, J., Harman, M., Ochoa, G., and Burke, E. Generic software subgraph isomorphism. In *Proceedings of the 4th International Symposium on Search Based Software Engineering (SSBSE '12) - Fast Abstracts* (2012).

[10] Wang, C.-X., Cui, Y.-A., Li, X., Chen, H., and Cui, D.-W. A novel ant colony system based on traditional chinese medicine theory. *International Journal of Computer Science and Network Security 6*, 5 (May 2006), 153.