Efficiency-Guided Method for Traveling-Salesman-Like Resource Allocation Problems

William H. Press*
Harvard University, Cambridge, Massachusetts
and
Curtis G. Callan Jr.†
Princeton University, Princeton, New Jersey

In resource allocation problems, it is possible to define the local efficiency of a small change, namely the ratio of change in value to marginal cost. This efficiency can be used to "guide" the exploration of an otherwise stochastic method like simulated annealing. An algorithm embodying this idea, termed efficiency-guided addition, subtraction, and permutation, (EGASP), is described for problems akin to the traveling salesman problem.

I. Introduction

WE discuss in this paper a method for approximately solving a class of resource allocation problems that are loosely based on the traveling salesman problem. The method and its variants are named EGASP, an acronym for efficiency-guided addition, subtraction, and permutation. The EGASP method is stochastic, and it is somewhat related to the method of simulated annealing.[1-3] More precisely, EGASP can be thought of as a substrategy for choosing random moves within the context of the simulated annealing method.

Alternatively, EGASP can be used independently of a simulated annealing "outer" strategy, equivalent to (as we will see) the case where the solution space is explored entirely at large temperature. For sufficiently simple problems, including that posed by the 1987 AIAA Design Challenge in Artificial Intelligence, EGASP alone is found to be about as good as EGASP with simulated annealing. We will discuss the relation between EGASP and simulated annealing further.

In this paper, it is useful to distinguish the following sequence of problems:

Problem A (traveling salesman). Given a set of cities, and a table of intercity airfares, find a closed path that visits all the cities at minimum cost.

Problem B (overworked salesman). Given a set of cities, a starting city, a table of intercity airfares, a value (reward) for each city, and a limited budget, find an *affordable* closed path that maximizes the sum of the values of the cities visited. (In the interesting case, the budget will not allow all cities to be visited.)

Problem C (overworked salesman with probabilistic costs). Same setup as problem B, but with intercity airfares that are probabilistic. Find a closed path that maximizes the summed city values, while having less than some specified probability of being over budget.

Problem D. Like problem C, but with additional complications: values of cities or airfares may depend on the order visited, additional path-dependent costs may be levied, etc.

The EGASP method applies most directly to problem B. We will see, however, that it can easily be applied to problems C and D through the use of additional, straightforward heuristics. Since the 1987 Design Challenge is in the class of problem D, we have incorporated these heuristics in the program there submitted, and will discuss them in this paper.

The key idea of which EGASP is based derives from the difference between problem A and problem B. Problem B has more structure than problem A. We want to exploit this additional structure to devise a stochastic algorithm that is "smarter" than "dumb" simulated annealing (already known to be successful at solving problem A).

In problem A, the goal (to visit all cities) is fixed, whereas the resource (budget) is open-ended. The object is to minimize use of the open-ended resource. Since the goal is achieved only by visiting the *last* city, there is no natural "local" sense in which an incremental expenditure of resource on some intermediate travel segment can be weighed against an incremental increase of goal.

By contrast, in problem B, the goal (accumulation of city values) is open-ended, and the resource (money) is also incrementally variable, though bounded from above by the budget limit. The object is to maximize the goal. Problem B is a resource allocation problem, whereas problem A is a straight optimization problem. In problem B there is a natural local measure of efficiency: the ratio of an incremental expenditure of resource to the incremental added value of a new city added to an existing tour. It is this local efficiency measure that EGASP seeks to exploit. In the simplest case of problem B, EGASP also uses the fact that both cost and value are strictly additive as cities are accumulated. The extension of EGASP to problems C and D involves casting those problems in an approximately additive form.

II. Terminology and Outer Program Logic

We standardize terminology as follows: A *path* is an ordered list of N cities, not necessarily distinct, beginning and ending with the specified starting city. A path contains $N-1$ connecting *links*. A *segment* is any subpath of a path, that is, M consecutive cities and their $M-1$ consecutive links. The first occurrence of a city in a path is *marked*, whereas subsequent occurrences are *unmarked*. The *score* of a path is a sum of values for each marked city. The *cost* of a path is a sum of terms, with individual coefficients for each link, marked city, and unmarked city. (We consider the case of probabilistic costs below.)

A *move* consists of any of several alterations of a path:

1) A city can be *added* between two existing cities in the

Received May 29, 1987; presented as Paper 87-2334 at the AIAA Guidance, Navigation, and Control Conference, Monterey, CA, Aug. 17-19, 1987; revision received Nov. 10, 1987.

*Department of Physics and Harvard-Smithsonian Center for Astrophysics.
†Department of Physics.

path; this involves the deletion of one link and addition of two new links.

2) A city can be *subtracted*, which is the inverse process.

3) A city can be *permuted*, i.e., moved from one location in the path to another; this involves the deletion of three links, and their replacement by three different links.

4) An entire segment can be deleted; this involves the deletion of M consecutive cities in the path, the deletion of $M + 1$ consecutive links, and the addition of one new link.

Other moves are logically possible, but we have not found them to be necessary.

An efficiency e can be calculated for any move or contemplated move. The efficiency is defined by the changes in score and cost that the move effects:

$$e \equiv \Delta(\text{score})/\Delta(\text{cost}) \tag{1}$$

Evidently, a move and its inverse have the same efficiency. When the efficiency is positive, a large value is a generally better move for an addition, whereas a small value is a generally better move for a subtraction. When a move's efficiency is zero or negative, the move is termed either *motherhood* (increased score at decreased cost) or *forbidden* (decreased score at increased cost), depending on the sign of the numerator in Eq. (1). Figure 1 illustrates these relationships.

We can now describe the outer program logic (outer loop) of EGASP. The outer loop is a simple alternation between an "uphill" (unfavorable) procedure, the segment-delete move already described, and a "downhill" (favorable) procedure termed *regeneration* and described in the next section. The alternation begins with an initial allowed path, which may be the trivial path with $N = 2$, starting and ending on the same city. The segment-delete and regeneration procedures are then performed alternately for as long as time will allow. The path is scored before each segment deletion is made, and the highest-scoring path to have occurred is output as the final answer.

Segment deletion, in general, decreases a path's score. It is an uphill (unfavorable) move. Thus, there is opportunity here, in the outer loop, for a simulated annealing supervisory strategy to be implemented: A random segment is proposed for deletion. The implied decrease in score ΔE is computed. The move is either taken, with probability $\exp(-\Delta E/T)$, or rejected, with the complementary probability. If the move is rejected, another random segment is proposed for deletion. The "temperature" T is started large and slowly decreases according to some annealing schedule.

If T is very large, then a proposed segment is always deleted. This is the limit of an EGASP-only strategy, without simulated annealing. We have found that, in the simple case of 1987 AIAA Design Challenge, this method is indistinguishable from full-fledged simulated annealing. Note that this method does *not* correspond to simulated annealing at zero temperature, i.e., rapid quenching. That would be a strategy that makes only a single regeneration and then quits, which we have found to perform poorly. In some problems unrelated to the AIAA Design Challenge, we have found that simulated annealing in the outer loop does result in a significant increase in performance.

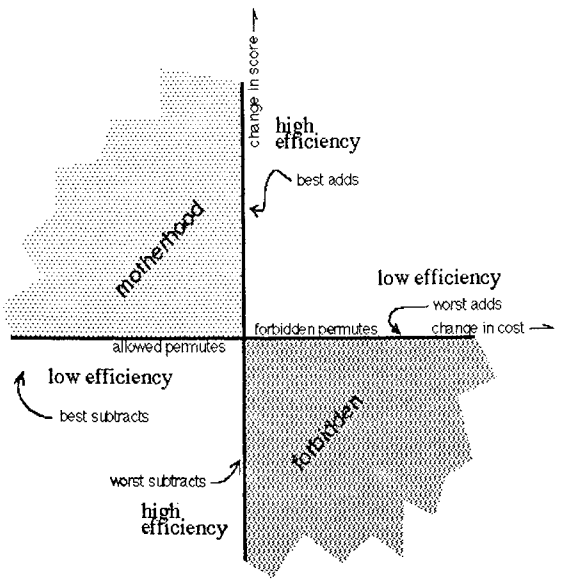

Fig. 1 Efficiency of the various moves used.

III. The Regeneration Procedure

The regeneration procedure always increases a path's score. It is a favorable move which, in the spirit of simulated annealing, will always be taken. Among all such favorable moves, the regeneration procedure seeks to use efficiency [Eq. (1)] as a guide toward particularly fruitful directions.

The regeneration procedure alternates among single-city adds, subtracts, and permutes (ASP). The general idea is to add a new city to the path, then—with probability p_{sub} usually ≈ 0.25—subtract an old city from the path, then add another city, and so on. If no city can be added to the path (within budget), then a failure counter is incremented, a favorable permute is attempted, and a subtraction is made. When the failure counter reaches some maximum value (usually 3), then a sequence of final forced adds is made, the regeneration terminating at the first failure.

We have not yet mentioned which city is added, where it is added, and which city is subtracted in these moves. That is where the efficiency guiding (EG) is brought to bear. For an add, all cities are tested at all possible sites. The efficiency of adding each city at its best (most efficient) possible site is saved. One city is then selected stochastically, with a weighting proportional to some monotonically increasing function of the efficiency. Empirically, we find that

$$\text{addition selection probability} \propto e^2 \tag{2}$$

works better than other simple functions.

For a subtraction, the efficiency of subtracting each city is calculated, and a city is selected stochastically with weighting

$$\text{subtraction selection probability} \propto e^{-2} \tag{3}$$

There is likely no special significance to the exact functional forms of Eqs. (2) and (3). It *is* important, however, that a stochastic algorithm be used, so that the regeneration procedure is able to sample a large variety of possible additions. Also, the mingling of add and subtract moves is found to be much superior to a sequence of add moves alone. This gives a kind of "minisimulated annealing" in the regeneration, with the efficiency guiding taking the place of the usual Metropolis algorithm, and with the saturation of the failure counter being a kind of one-step annealing schedule. However, the analogy is not mathematically perfect.

Except for a few small details, this completes the description of EGASP for the simplest case of problem B.

IV. Problems C and D

We here describe variations of the EGASP method that make it applicable to the peculiarities of the 1981 AIAA Design Challenge.

Airfares in the Design Challenge are probabilistic: on each link, there is a finite probability that the salesman will have to fly first-class. The overall constraint is not to be strictly below budget in all cases, but is rather to have no less than a specified probability of being under budget.

The exact calculation of a path's probability of being under budget is expensive, growing with number of cities in the path as 2^{N-1}. It is desirable that this calculation be performed only rarely. We do it only in the outer loop of EGASP, and only when a path is found that purports to be better than the best

saved path thus far. The new path will have been estimated (by the method described below) to be within the global probability constraint. If, in fact, it is, then it replaces the stored best path. If it turns out not to be, it is discarded. The number of full probability calculations increases only logarithmically with running time.

It is worth mentioning that, when an exact probability determination is required, it is not necessary to calculate out the whole decision tree of 2^{N-1} probabilities: upper and lower bounds on cost can be assigned to each tree node. If, at any stage of evaluation, a node's lower cost bound is over budget, then all branches below that node can be omitted from the calculation, likewise if a node's upper cost bound is under budget.

The "fast" way of estimating that a path is likely to be within the global probability constraint, and of assigning a quasilinear Δ(cost) to individual moves, is as follows.

As links are added to or subtracted from the path, a running sum of both the sum of the costs and the sum of the square of the costs is maintained. This allows both the expectation value and standard deviation of a path's cost to be calculated by the equations

$$\mu = \sum_i C_i(1+qf) \tag{4}$$

$$\sigma = f\sqrt{q(1-q)}\,\sqrt{\sum_i C_i^2} \tag{5}$$

where the "coach" cost of link i is C_i, the probability of a first-class increment is q, and the incremental cost factor is f.

The rough approximation is then made that the probability distribution of the total cost is Gaussian. If p is the allowed probability of being over budget, then the cost C of a path is taken to be the cost at a point on the Gaussian that is exceeded only with probability p. The equation for this is

$$C = (1+qf)\sum_i C_i + \alpha f\sqrt{q(1-q)}\,\sqrt{\sum_i C_i^2} \tag{6}$$

where α is the number of standard deviations for one-sided Gaussian tail probability p, approximately given by

$$\alpha = t - \frac{2.30753 + 0.27061t}{1 + 0.99229t + 0.04481t^2} \tag{7}$$

with

$$t \equiv \sqrt{-\ell n\, p^2} \tag{8}$$

(see Ref. 4). The quantity ΔC is now used for Δ(cost) in calculating the efficiency of all proposed moves.

In practice it is found useful to adjust α adaptively from its theoretical value (Eq. 7). This is done in the outer program loop whenever an exact probability of being over budget is calculated. If that probability is greater than p, then α is incremented by a small amount; if it is less than p, α is slightly decremented. This completes the description for problem C.

Problem D is a grab bag of additional complications. The most worrisome are those that couple the value of a move to the presence or absence of an earlier or later move, since these make the notion of local efficiency less powerful. In the AIAA Design Challenge, the value of one particular city (LAX) is lost unless another particular city (BOS) is included at some later position in the tour.

It is straightforward to keep track of the bookkeeping so that changes in value and cost are accurately known for any move involving the two particular cities. This does not, however, completely solve the problem: if BOS is not in the path, LAX will never be added, since it has zero value. Thus, BOS also does not get its proper weighting, because its value is not augmented by the potential added value of LAX.

We overcome these problems by introducing special rules that "know about" the additional constraints. In the above example, the only special rule introduced is to reserve a certain (small) fraction of all add moves for BOS, whether that city appears to deserve it or not. Once BOS has been added "on spec," as it were, the next add move automatically gives LAX an appropriately high value for certain proposed points of insertion; it thus frequently gets added. Once LAX is added, BOS is automatically stabilized, since the change in value for removing it would be large. If LAX does not happen to be added, or cannot be, then the next subtract move almost always deletes BOS, since it has substandard efficiency.

This is a special case of the general idea, that the stochastic moves of the regeneration process must be sufficiently robust so as to explore move correlations that are made important by any additional constraints in the problem. When such robustness is achieved, then the EGASP method goes through unchanged.

Acknowledgment

This work was supported in part by internal R & D funds of the JASON Program Office, MITRE Corporation.

References

[1]Kirkpatrick, S., Gelatt, C. D., and Vecchi, M. P., *Science*, Vol. 220, 1983, p. 671–680.

[2]Kirkpatrick, S., *Journal of Statistical Physics*, Vol. 34, 1984, p. 975–986.

[3]Press, W. H., Flannery, B. P., Teukolsky, S. A., and Vetterling, W. T., *Numerical Recipes: The Art of Scientific Computing*, Cambridge University Press, New York, 1986, p. 326.

[4]Abramowitz, M. and Stegun, I. A., *Handbook of Mathematical Functions*, National Bureau of Standards, Washington, DC, 1964, p. 933.