Greedy Heuristics with Regret, with Application to the Cheapest Insertion Algorithm for the TSP

Refael Hassin* Ariel Keinan †

Abstract

We considers greedy algorithms that allow partial regret. As an example we consider a variant of the cheapest insertion algorithm for the TSP. Our numerical study indicates that in most cases it significantly reduces the relative error, and the added computational time is quite small.

Keywords: Cheapest insertion heuristic, greedy algorithm with regret, traveling salesman problem

Introduction

Many common heuristics for combinatorial optimization problems are considered as *greedy*. These are constructive heuristics, designed to produce solutions of reasonable quality without investing the time needed to compute better solutions by other methods, such as meta-heuristics, based on iterative improvements. Greedy algorithms make irrevocable decisions about the construction of a solution, based on local considerations such as preferring the choice that gives immediate best reward or minimum cost. In this note we examine the possibility of improving the quality of a greedy algorithm by allowing it to reconsider decisions made in past steps. We maintain however the greedy spirit of the algorithm by allowing only *limited regret*, so that the result is still a fast constructive algorithm.

Hassin and Levin [2] applied the idea of a greedy algorithm with limited regret to the set covering problem, by allowing the reversal of an earlier decision to include a given set in the solution, if its present impact contradicts the greedy choice. In other words, a decision that wouldn't have been made given the current costs, can be reversed. They proved that such a modification improves the worst case error for the weighted set covering problem.

This note presents a preliminary study of the possible practical benefits associated with incorporating limited regret into greedy heuristics. In this study, we apply the idea to a well known greedy type algorithm for the traveling salesman problem, and run a computational experiment to check its usefulness.

It should be emphasized that our goal in this study is to examine the effect of allowing regret in the greedy approach. A variant that may be competitive with known best heuristics should incorporate further features such as randomization and repeated application, for example as suggested by Brest and Žerovnik [1]. However, we chose to apply the idea to the generic cheapest insertion algorithm so that its effect can be assessed without the need to calibrate the parameters of a more sophisticated algorithm.

We find that indeed incorporating limited regret considerably reduces the average relative error of the algorithm. Specifically, we have applied the algorithm to all TSPLIB undirected problems with less than 4000 vertices and obtained an average typical reduction of the relative error from 16.4% to 11.8%. The added running time is about 70%.

*Department of Statistics and Operations Research, Tel-Aviv University, Tel-Aviv 69978, Israel. Tel: +97236409281 Email: hassin@math.tau.ac.il

†Department of Statistics and Operations Research, Tel-Aviv University, Tel-Aviv 69978, Israel. Email: keinan.ariel@gmail.com

This result indicates that such ideas are potentially useful for a variety of other greedy type algorithms, and encourages further theoretical and computational study.

1 The algorithm

Let $G = (V, E)$, $V = \{v_1, \ldots, v_n\}$, be a complete edge-weighted undirected graph, with $V = \{1, \ldots, n\}$. Denote by c_{ij} the length of $(i, j) \in E$.

A tour is a Hamiltonian cycle in G, that is, a simple cycle with n vertices. A partial tour is a pair of parallel edges, or a simple cycle in G with at most $n - 1$ vertices. For a partial tour T and a vertex $k \notin T$ we define insert(T, k) to be the cycle obtained by deleting an edge $(i, j) \in T$ and inserting instead the two edges (i, k) and (k, j). We say that k is *inserted* into T. We define by $c(T, k)$ the length increase caused by the insertion of k into T, that is $c(T, k) = c(\text{insert}(T, k)) - c(T)$.

The *Cheapest Insertion* algorithm is a well known heuristic for the traveling salesman problem [3]. Moreover, when G is undirected and the edge lengths satisfy the triangle inequality, Rosenkrantz, Stearns, and Lewis [4] proved that it returns a solution of length at most twice the optimal. The algorithm is described in Figure 1. In our version, the initial partial tour is a minimum cost 2-edge cycle. Alternatively, it could start with an arbitrary vertex as the initial partial tour.

Cheapest Insertion
input
A weighted graph $G = (V, E)$.
begin
$k, l := \arg\min\{c_{ij} + c_{ji} : i, j \in V, i \neq j\}$.
$T := (k, l, k)$.
while *T is a partial tour*
$v := \arg\min\{c(T, k) : k \notin T\}$.
$T := \text{insert}(T, v)$.
end while
end *Cheapest Insertion*

Figure 1: The Cheapest Insertion algorithm

For a partial tour T and a vertex $k \in T$ we define delete(T, k) to be a shortest cycle obtained by deleting from T the two edges incident with k, say (i, k) and (k, j), and replacing them by (i, j). We say that k is *deleted* from T and mark the resulting subtour by delete(T, k). Thus, deletion is the inverse operation of insertion. We define by $c^-(T, k)$ the length reduction caused by the deletion of k from T, that is $c^-(T, k) = c(T) - c(\text{delete}(T, k))$.

We propose a modification of the cheapest insertion algorithm, which we call *cheapest insertion with regret*. The algorithm is described in Figure 2. At each step, we compare the cheapest next insertion and the maximum length reduction caused by deleting a vertex (which is not one of the subtour vertices participating in the insertion) from the current subtour.

Note that the deletion operation does not necessarily reverse a previous insertion, since it may be that the deleted vertex has been added with different edges than those which are now deleted. Still we might think about the deletion step as a *regret* in the sense that we allow giving up part of the subtour if we see that it costs more than the new insertion.

A further step in the direction of the new algorithm would be to allow more than a single deletion, however this might result in long series of insertion and deletions and even in lack of convergence of the algorithm caused by cyclic sequences of subtours, as we illustrate in the next

section. In contrast, the version we propose converges since at each step we either obtain a subtour with an additional vertex, or we get a strictly shorter subtour with the same number of vertices as before.

Cheapest Insertion with Regret
input
A weighted graph $G = (V, E)$.
begin
$k, l := \arg\min\{c_{ij} + c_{ji} : i, j \in V, i \neq j\}$.
$T := (k, l, k)$.
while *T is a partial tour*
$v := \arg\min\{c(T, k) : k \notin T\}$.
$(i, j) :=$ *the edge of* T *replaced in* $\text{insert}(T, v)$.
$u := \arg\max\{c^-(T, k) : k \in T \setminus \{i, j\}\}$.
if $c(T, v) \geq c^-(T, u)$
then
$T := \text{insert}(T, v)$.
else
$T := \text{insert}(\text{delete}(T, u), v)$.
end if
end while
end *Cheapest Insertion with Regret*

Figure 2: The Cheapest Insertion with Regret algorithm

2 An example

Figure 3 (a)-(f) illustrates how the algorithm works. We consider six points in the plane at locations $(0, 3)$, $(0, 4)$, $(1, 0)$, $(1, 1)$, $(1, 3)$, and $(2, 3)$, with the Euclidean distances. The algorithm starts with a shortest 2-edge cycle on the points at $(1, 0)$ and $(1, 1)$, and then inserts the point $(1, 3)$ reaching illustration (a). At this instant, the best insertion is of $(2, 3)$ costing $\sqrt{10} - 2 \approx 1.16$. This insertion allows only one deletion, of $(1, 1)$ of value $0 < 1.16$ and thus the deletion is not done. The subsequent iterations are illustrated in the figure, showing also the relevant $c(T, k)$ values for non-cycle points and $c^-(T, k)$ values for cycle points. An interesting step is the passage from (b) to (c) which involves an insertion and a deletion.

A natural extension of the algorithm allows for several deletions while executing an insertion step, if all of these deletions do not interfere with the insertion and have higher values. We have tried this version and found out that in some cases it gives better results, but in others it tends to cycle. The latter phenomenon is not restricted to pathological cases but does happen often. To illustrate the possibility of cycling we slightly modify our example and assume that the upper left point is slightly higher, at $(0, 4 + \epsilon)$ for some small $\epsilon > 0$. The first cycles are as in illustrations (a)-(c). However, at this instant, there is a second deletion with value higher than the planned insertion, that of deleting the bottom point of the cycle, with value $2.47 > 1.24$. Performing this deletion, the algorithm moves to the cycles given in (d')-(f') and then back to (c).

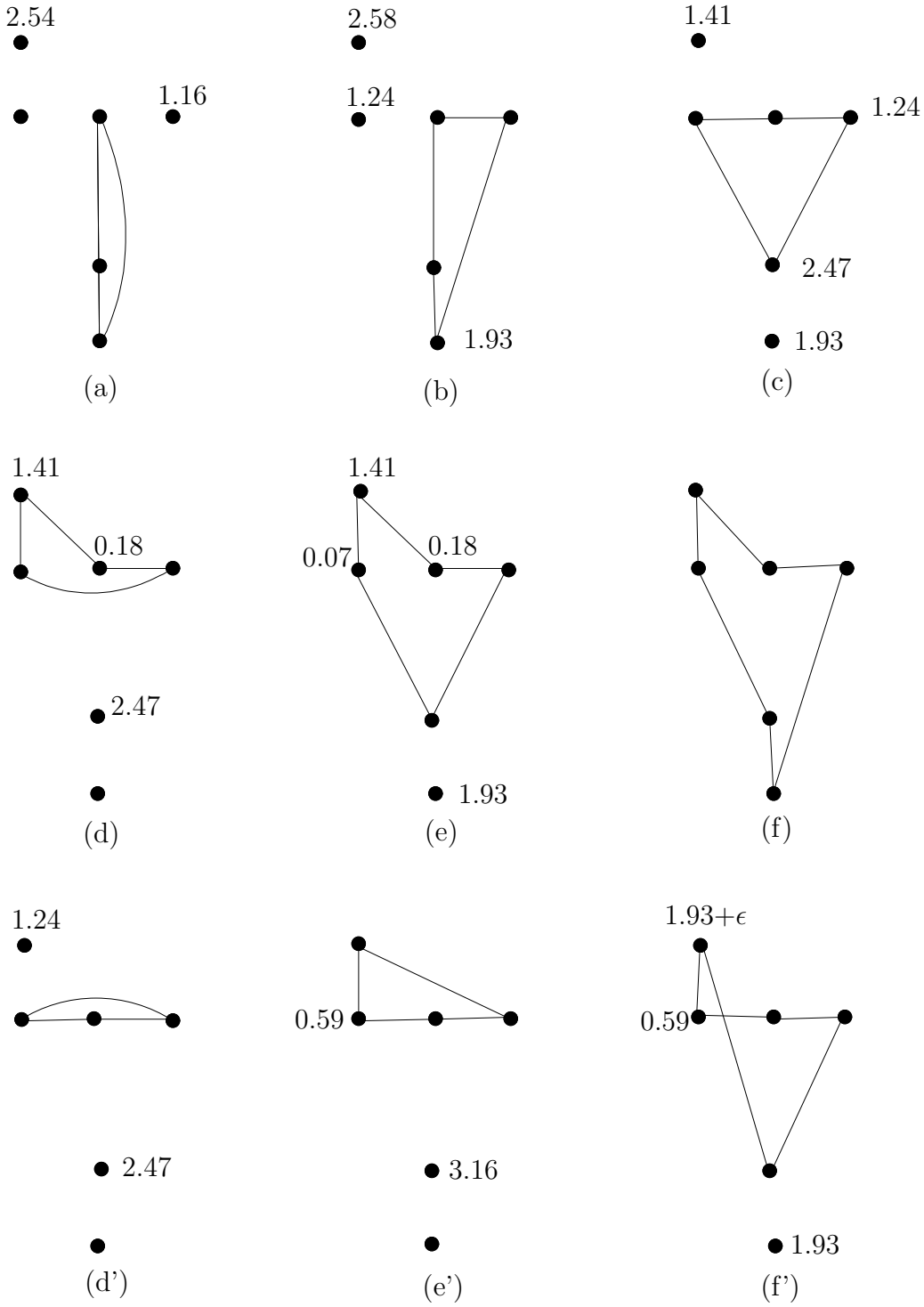

Figure 3: Example

3 Computational Results

We have tested the cheapest insertion with regret algorithm on the test problems in the TSPLIB (http://elib.zib.de/pub/mp-testdata/tsp/tsplib/tsplib.html). The results for the symmetric instances are given in Table 1. The first column gives the problem's name including the number of vertices in the graph. The second column gives the type of the distance function, as explained in the TSPLIB site. The third column gives the error of the standard cheapest insertion algorithm over the optimal solution as given in TSPLIB. It is given in percents and rounded to the nearest integer. The fourth column gives the same results but when the cheapest insertion with regret algorithm is applied to the instance. We note that the average error obtained for the standard algorithm is 16.4% whereas when regret is allowed the average ratio decreases to 11.8%. The number of iterations in the standard algorithm is of course $|V|$, and when regret is allowed it goes up to approximately $1.7|V|$ reflecting the added iterations in which there is also a deletion.

References

[1] J. Brest and J. Žerovnik, "A heuristic for the asymmetric traveling salesman problem," *The 6th Metaheuristic International Conference, MIC2005*, 2005.

[2] R. Hassin and A. Levin, "A better-than-greedy approximation algorithm for the minimum set cover problem," *SIAM J. on Computing* **35** (2005) 189-200.

[3] D.S. Johnson and C.H. Papadimitriou, Performance guarantees for heuristics. Ch. 5 in The Traveling Salesman Problem, E.L. Lawler, J.K. Lenstra, A.H.G. Rinnooy Kan, and D.B. Shmoys, (editors), Wiley & Sons, Chichester (1985).

[4] D.J. Rosenkrantz, R.E. Stearns, and P.M. Lewis II, "An analysis of several heuristics for the traveling salesman problem", *SIAM J. Comput.* **6**, 563–581, 1977.

Problem	Type	CI	CIR
bayg-29	matrix	15	8
bays-29	matrix	5	5
dantzig-42	matrix	14	11
swiss-42	matrix	16	11
gr-48	matrix	16	7
hk-48	matrix	14	7
eil-51	euc	9	7
berlin-52	euc	19	16
brazil-58	matrix	16	11
st-70	euc	17	9
eil-76	euc	13	11
pr-76	euc	16	14
gr-96	geo	27	18
rat-99	euc	20	15
kroA-100	euc	19	12
kroB-100	euc	14	10
kroC-100	euc	22	16
kroD-100	euc	18	15
kroE-100	euc	15	9
rd-100	euc	16	10
eil-101	euc	14	10
lin-105	euc	18	11
pr-107	euc	19	15
gr-120	matrix	13	10
pr-124	euc	12	3
bier-127	euc	19	15
ch-130	euc	16	10
pr-136	euc	14	14
gr-137	geo	18	15
pr-144	euc	25	5
ch-150	euc	19	13
korA-150	euc	13	10
korB-150	euc	20	17
pr-152	euc	22	11
u-159	euc	20	11
si-175	matrix	4	3
brg-180	matrix	24	11
rat-195	euc	18	13
d-198	euc	12	7
korA-200	euc	20	12
korB-200	euc	22	16
gr-202	geo	16	14
ts-225	euc	26	14
tsp-225	euc	19	10

Problem	Type	CI	CIR
pr-226	euc	13	10
gr-229	geo	13	11
gil-262	euc	17	11
pr-264	euc	19	16
pr-299	euc	20	16
lin-318	euc	18	13
rd-400	euc	22	15
fl-417	euc	21	7
gr-431	geo	13	12
pr-439	euc	22	16
pcb-442	euc	20	12
d-493	euc	14	12
ali-535	geo	20	17
si-535	matrix	2	2
pa-561	matrix	24	20
u-574	euc	18	14
rat-575	euc	18	14
p-654	euc	17	10
d-657	euc	19	16
gr-666	geo	20	15
u-724	euc	21	15
rat-783	euc	18	15
pr-1002	euc	17	15
si-1032	matrix	0	1
u-1060	euc	21	17
vm-1084	euc	16	12
pcb-1173	euc	22	15
d-1291	euc	16	14
rl-1304	euc	24	18
rl-1323	euc	24	16
nrw-1379	euc	17	12
fl-1400	euc	16	9
u-1432	euc	12	8
fl-1577	euc	18	15
d-1655	euc	17	14
vm-1748	euc	19	15
u-1817	euc	16	13
rl-1889	euc	23	19
d-2103	euc	10	5
u-2152	euc	17	12
u-2319	euc	9	4
pr-2392	euc	21	19
pcb-3038	euc	17	14
fl-3795	euc	12	10

Table 1