Proceedings of the 2009 IEEE International Conference on Systems, Man, and Cybernetics
San Antonio, TX, USA - October 2009

Tabu Split and Merge for the Simplification of Polygonal Curves

Gildas Ménier
VALORIA, Université de Bretagne Sud
Université Européenne de Bretagne
France
gildas.menier@univ-ubs.fr

Pierre-François Marteau
VALORIA, Université de Bretagne Sud
Université Européenne de Bretagne
France
pierre-francois.marteau@univ-ubs.fr

***Abstract*— A Tabu Move Merge Split (TMMS) algorithm is proposed for the polygonal approximation problem. TMMS incorporates a tabu principle to avoid premature convergence into local minima. TMMS is compared to optimal, near to optimal top down Multi-Resolution (TDMR) and classical split and merge heuristics solutions. Experiments show that potential improvements for crudest approximations can be obtained. The evaluation is carried out on 2D geographic maps according to effectiveness and efficiency measures.**

***Keywords*— Polygonal approximation, Tabu search, Split-and-Merge, top down multi resolution, dynamic programming.**

I. Introduction

Polygonal curve approximation has been widely studied in the past to scale up time consuming applications such as graphic display, contour detection or time series data mining. If we apprehend a discrete curve as a multidimensional vector, polygonal approximation can be seen as a dimension reduction technique that relates also to multidimensional scaling.

Following polygonal approximation is an optimization problem that can be tackled along two angles:

- *Min-ε* problem: Given a polygonal curve S_c having N segments, find an approximation A_c having K segments such that the maximal approximation error ε is minimized.
- *Min-#* problem: Given a polygonal curve S_c having N segments, find an approximation A_c with the minimum number of segments K so that the maximal approximation error does not exceed a given tolerance ε.

Most of the proposed algorithms developed to solve these problems belongs either to graph-theoretic approaches [1,6,8,10,11,12], dynamic programming [5,7] or to heuristic approaches [2,4,12,13,15]. Optimal solutions based on dynamic programming principle exist, nevertheless, there complexities are proved to be $O(K.N^2)$ leaving space for much faster but sub-optimal solutions.

Finding fast algorithms as near-to-optimality as possible for long input curves is still an open challenge leading to fruitful applications. In [10] for instance, author proposed a fast and dirty filtering approach dedicated to time series retrieval whose efficiency is highly correlated with the quality of polygonal approximations.

Recently, we have proposed a top town multiresolution algorithm (TDMR) [9] that solves the problem of polygonal curve approximation in linear time complexity *O(N)*. This algorithm ensures near to optimal solutions between each two successive levels of resolution, but, as we descend the resolution levels, the approximations depart further from optimality. The only other known algorithm showing a linear complexity is the Douglas-Peucker algorithm [2,4]. It is faster than TDMR but provides approximations that are much farther to optimality than the ones provided by TDMR. In this paper, we explore heuristic strategies based on three elementary operations (*merge, split* and *move*) associated to a Tabu search principle and try to evaluate how these strategies could boost suboptimal solutions such as TDMR, Merge-L2 [12] or Douglas Peucker [2] heuristics.

The second section of the paper states the problem definitions and introduces the three elementary operations at the basis of the heuristics we will detail into the third section of the paper. The fourth section presents and comments the experiments we have carried out on a set of 2D geographic maps; we conclude the paper and suggest some perspectives in the final section of the paper.

II. Problem definitions and the three elementary operations

We attempt to correct an approximation curve A_c of k points approaching a (source) curve S_c of n points ($k << n$) by the mean of elementary transformation operators: *move*, *merge* and *split*. To simplify the search space, and following the common definitions of the *Min-ε* and *Min-#* problems, we take approximation points among the points of the target curve: A_c is therefore a sorted list of references (index) of points in S_c.

Let S_c[i] be the ith point of the curve S_c.

Let A_c[] be the reference to a point in Sc – the ith element of A_c is the A_c[i]th point of S_c, i.e. the point $S_c[A_c[i]]$ (Fig. 1).

978-1-4244-2794-9/09/$25.00 ©2009 IEEE

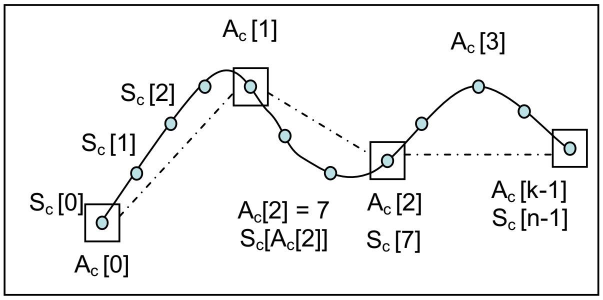

Figure 1. The reference curve S_c and an approximation A_c

Let $S_c[a,b]$ be the sub curve between the a^{th} point and the b^{th} point of the curve S_c.
Let $A_c[a,b]$ be the sub curve between the a^{th} point and the b^{th} point of the curve A_c.

A. move operator

With the aim of minimizing the *Min-ε* criterion, the *move* operator slides a potentially misplaced point thru the set of curve's indexes according to least surface error gradient direction. The error is computed as follows:

For the approximation point i_a (coding the i_a^{th} element of the curve A_c), an evaluation of the surface between the sub curve $S_c[A_c[i_a-1], A_c[i_a]]$ and the sub curve of $A_c[i_a-1, i_a]$ is computed as $E_{left}(S_c, i_a)$ summing the Euclidian distances between the points of S_c and the approximation line provided by $Ac[i_a-1, i_a]$ (Figure 2).

$$E_{left}(i_a) = \sum_{j=Ac[i_a-1]}^{j<Ac[i_a]} d_{eucl}(A_c[i_a-1, i_a], S_c[j])$$

$d_{eucl}(D,P)$ stands for the Euclidian distance between a point P and a line D.

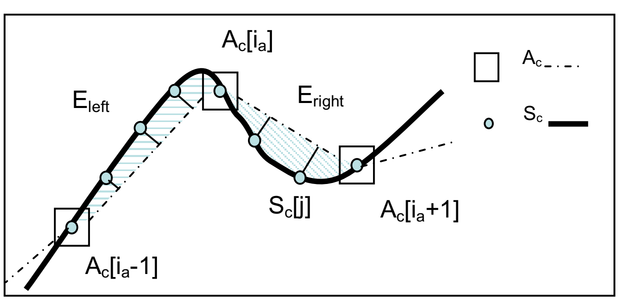

Figure 2. Gradient of the errors for the *move* operation

Likewise,

$$E_{right}(i_a) = \sum_{j=Ac[i_a]}^{j<Ac[i_a+1]} d_{eucl}(A_c[i_a, i_a+1], S_c[j])$$

Let $E_{left/right}(i)$ be the surface error balance supported by the i^{th} point:

$$E_{left/right}(i_a) = | E_{left}(i_a) - E_{right}(i_a) |.$$

The main idea here is to slide the i_a^{th} point (in the S_c indexes space) to balance the left and right errors. If $E_{right}(i_a) > E_{left}(i_a)$ then the i_a^{th} point is moved by one position right in the S_c indexes space (respectively left if $E_{right}(i_a) < E_{left}(i_a)$). The displacement is constrained by the segment boundary points i_a-1 and i_a+1 (Alg. 1).

```
OPERATOR move
BEGIN
        foreach point i in Ac
                compute Eleft/right (i)
        end foreach
        select the point imax having the maximum Eleft/right (i)
        if Eleft(imax) < Eright(imax)
        then
                if (Ac[imax-1]<Ac[imax] -1)
                then
                        Ac[imax] := Ac[imax]  -1
                         // slides i to the left
                end if
        else
                if Eleft(imax) > Eright(imax)
                then
                        if (Ac[imax+1] > Ac[imax] +1)
                        then
                         Ac[imax] = Ac[imax]  +1
                         // slides i to the right
                        end if
                end if
        end if
END
```

Algorithm 1

One execution of the operator *move* slides only one point - the point of A_c having the worse $E_{left/right}(i)$ (thus being the most unbalanced point on the A_c curve regarding the local surface error).

B. split operator

The *split* operator (Alg. 2) finds one of the point i_c in S_c that maximises d_{eucl} (A_c, $S_c[i]$) and inserts a new point in A_c to correct this error – thus decreasing (eventually not strictly) the *Min-ε* value.

OPERATOR *split*
BEGIN
foreach point i_c in S_c
compute $d_{eucl}(A_c, S_c[i])$
end foreach
select the point ic_{max} with the maximum distance
insert a new point i_{new} in A_c
that minimizes
$d_{eucl}(A_c[i_{new}-1, i_{new}+1], S_c[ic_{max}])$
END

Algorithm 2

OPERATOR *merge*
BEGIN
foreach element i_a in A_c
compute $R_p(i_a)$
end foreach
select the element i_a having the higher removal priority R_p
remove it
END

Algorithm 3

C. *merge operator*

The *merge* operator removes an element from A_c, thus merging the two adjacent approximating segments. Since the ratio *k/error* is important, the point to remove should be the point contributing the least to the global error – *ie*, it is important to discard first the points which removal increases the least the global error.

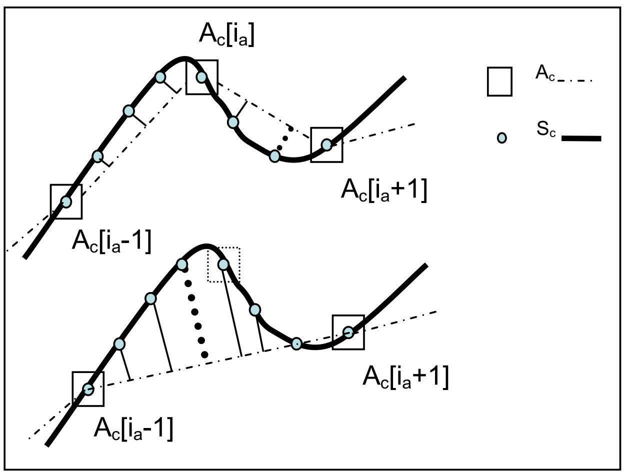

Figure 3. Application of the merge operation on element i_a.

The removal priority R_p for an element i_a is computed as follows : The error criterion *Error* (*Min-ε* for instance) is computed on the segments $[A_c[i_a-1], Ac[i_a]]$, $[A_c[i_a], Ac[i_a+1]]$ (Figure 3) and if the element $A_c[i_a]$ were to be discarded, on the segment $[A_c[i_a-1], Ac[i_a+1]]$, the two segments $[A_c[i_a-1], Ac[i_a]]$ and $[A_c[i_a], Ac[i_a+1]]$ are merged.

For instance, if the *Min-ε* error is used:

$$R_p(i_a) = \mathrm{Max}(\mathrm{Error}_{min\text{-}e}[i_a-1, i_a], \mathrm{Error}_{min\text{-}e}[i_a, i_a+1]) / \mathrm{Error}_{min\text{-}e}[i_a-1, i_a+1]$$

Alg. 3 describes the process :

D. *Refining the curve*

As introduced above, we attempt to correct an existing approximation curve A_c of *k* points approaching a (source) curve S_c of *n* points by the mean of elementary transformation operators *move*, *merge* and *split* :

FUNCTION *Refine* (A_c, S_c)
// Refines the approximation A_c (of k elements) of the curve S_c (n points)
BEGIN
repeat 2*k loops of
move();
merge();
split();
end repeat
END

Algorithm 4

Unfortunately, a sequence of *move*, *split* and *merge* does not ensure that the error criterion will strictly decrease. In fact, the experiments (see Fig. 4) show a sequence of global error changes similar to those existing in incremental learning (such as Back Propagation Networks, or to a least amount in Genetic Algorithm) – *ie* in stabilizations and step breaks: this suggests that the optimization of the locations of the points is not a gradient process but rather a complex organizing mechanism necessary to explore the parameters space.

It is then necessary to keep aside the best solution found and to expect the next iterations to eventually improve this solution.
In the simple algorithm described above (Alg. 4), we decided to perform a number of loops proportional to the *k* elements of the approximation since we think that, in the ideal configuration, each element of the approximation should undergo – a least once - a *move*/*merge*/*split* operator.

Experimentally, we observe that *k* loops is a minimum limit to achieve the best expectable error: experimentally, more than *2*k* loops doesn't increases the performances so far.

During its exploration of the parameter's space (as a vector of k indexes in the S_c curves), this algorithm seems to fall in gradient wells: Fig. 4 exhibits oscillating behaviours that prevents the search to go on: it seems that the sequence of *move*/*merge*/*split* can lead to configurations that occur sequentially after t loops.

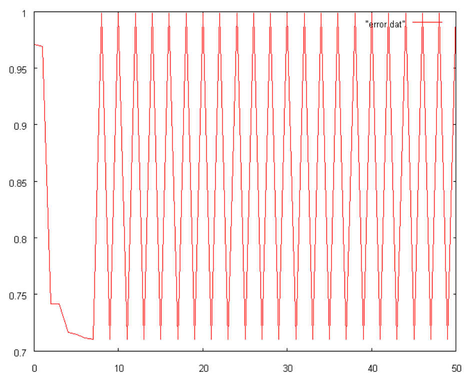

Figure 4. Curve of n = 49344 points with k = 50 (best solution min-e = 0.716) – No Tabu involved.

This suggests that the same configuration of elements in A_c may be scanned more than once. Keeping the n^{th} last configurations of A_c^n and preventing (temporary) the algorithm to re-scan these previous solutions is somehow related to a search space strategy called Tabu Search [8].

In this example, we assumed that the cost to store and compare a new solution of A_c (of k points) to the t previous solutions would be important. Mostly, it is the recurring choice of the points to *move*, *merge* and *split* that leads to an endless evaluation of sequences of already-seen configurations.

Therefore, we propose to introduce some Tabu search principles through the management a list of 'recently points chosen to perform *move*/*merge*/*split'* and prevent the operator from picking (again) one of the tagged element – at least as long as these elements are enqueued in this FIFO list.

Restricting the search by the choice of one index point segments the search space in classes of approximation curves containing (or not) a specific point: this could be very restrictive at a first glance, but this strategy speeds up the search by selecting the curve containing the best / worse point in regard to the *Min-ε* criterion.

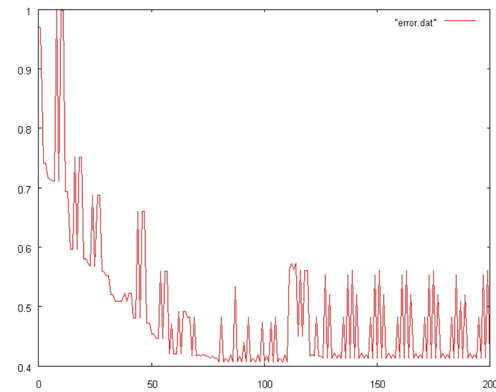

Figure 5. Tabu with 5 elements (best solution min-e = 0.412)

For a Tabu list of size t_b elements, not only this strategy seems to prevent the rapid oscillating behaviour under t_b loops, but also seems to increase the search speed.

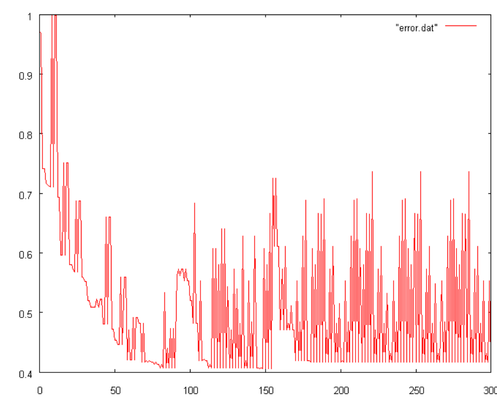

Figure 6. Tabu of size 20 (best solution min-e = 0.409)

Increasing the size of the Tabu list above 10-15 elements decreases the performance of the algorithm without significant better results (mostly under 10^{-4} for the maximum error).

III. Approaches

A. RSDP: Reduced Search Dynamic Programming

The reference algorithms for curve approximations for the *Min-ε* criterion are mostly based on dynamic programming [12]: they usually provide the minimal error at a computational cost of $O(n^2)$. It is possible to reduce this complexity, constraining the search when near-optimal solutions are acceptable – lowering the computational cost to $O(n^2/k)$: in the following experiments, RSDP stands for *Reduced Search Dynamic Programming* [7].

B. MR : Multi Resolution

In [9], we introduced a top-down multi resolution algorithm TDMR designed to compute iteratively nested approximations with a complexity (at the best case) of $O(n)$: it features sequential processing of RSDP-like processing and outputs a multiresolution solution to the approximation problem.

C. I-TMMS : Refine with equidistant initialization

This processing involves the *refine* function (described above) starting with a first curve of k points to correct: each k_i point is initially set at equidistant position on the S_c curve (rounded to the nearest index). *2*k* loops are performed.

D. SPLT : Split

Starting with an initial curve of $k_0=2$ points (the first and last point of S_c), this algorithm performs *k-2 split* operations to reach the final k elements for A_c.

E. MRG : Merge

Likewise, this algorithm starts with the complete curve S_c – as the full collection of indexes for A_c – and decimates iteratively the points (by the mean of *merge* operators) until the number of remaining elements reaches k elements in A_c.

F. MR/TMMS

A multiresolution process (MR) is performed [9], *refined* by the tabu *move*/*merge*/*split* (TMMS) sequence of operators.

G. SPLT/ TMMS

The SPLT (split) is performed, followed by the *refine* (tabu *move*/*merge*/*split*) TMMS process.

H. MRG/TMMS

The MRG (merge) is performed, followed by the *refine* (tabu *move*/*merge*/*split*) TMMS process.

IV. EXPERIMENTS

The experiments have been performed on 10 curves S_c depicting the costal maps of Western Europe. The 10 curves S_c have at least n = 8192 points.

We measure the fidelity of an approximation using the following formula:

$$F_{method} = E_{RDSP}/E_{method}$$

where E is the *Min-ε* error.

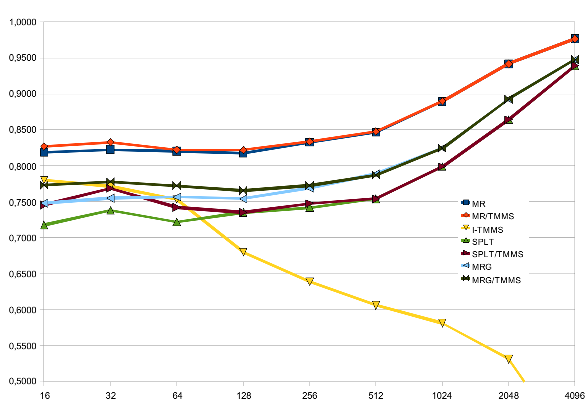

Figure 7. F_{method} for different values of k (number of points in A_c)

RSDP, that is near-optimal, is used as reference solution. Fig.7 shows the *Fidelity* for all the experimented methods. Basically, the TMMS procedure boosts the experimented methods for low k values. ISM performs quite well for values of $k << n$: it is only outperformed by MR and MR/TMMS : for $k>64$, ISM gives the worst results. MR/TMMS seems to improve marginally the error of MR. The TMMS procedure introduces a time cost that is measurable for all experimented methods in Fig.8.

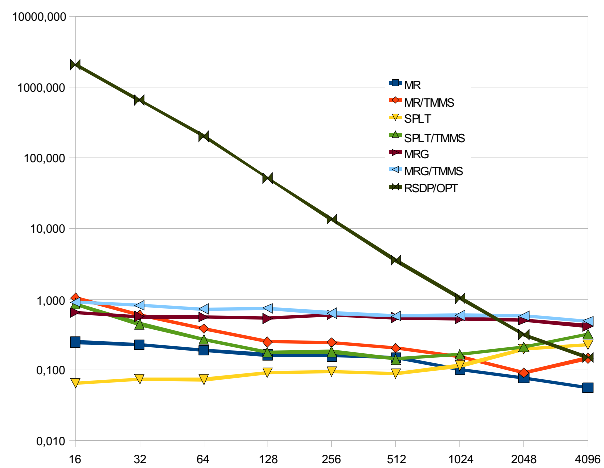

Figure 8. Evaluation for all experimented methods of computation time (in sec.) for different values of k.

V. CONCLUSION

We have introduced the use of *move*/*merge*/*split* operators using a Tabu-like selection to refine an existing approximation curve. We compared this approach with other sub optimal algorithms, namely *top down multiresolution*, *split* algorithm and *merge* algorithm. The TMMS procedure offers some boosting capability for the crudest approximations and could probably be used directly inside a multiresolution approach to improve the overall fidelity of the provided approximations a low level of resolution.

REFERENCES

[1] Chen D.Z., Daescu O.. “Space-efficient algorithms for approximating polygonal curves in two-dimensional space”. In Int. Conf. on Computing and Combinatorics, Lecture Notes in Computer Science, volume 1449, pp 4554, 1998.

[2] Douglas D., Peucker T., "Algorithms for the reduction of the number of points required to represent a digitized line or its caricature", The Canadian Cartographer 10(2), pp112-122, 1973.

[3] Glover F., Laguna M. “Tabu Search”. Kluwer, Norwell, MA, 1997.

[4] Hershberger J., Snoeyink J., "Speeding Up the Douglas-Peucker Line-Simplification Algorithm", Proceedings 5th Symp on Data Handling, 134-143. UBC Tech Report TR-92-07, 1992.

[5] Horng J.-H. “Improving fitting quality of polygonal approximation by using the dynamic programming technique”. Pattern Recognition Letters, 23:pp1657-1673, 2002.

[6] Imai H., Iri. M. “Polygonal approximations of a curve - formulations and algorithms”. Computational Morphology, pp 71-86, 1988.

[7] Kolesnikov A., Fränti P., "Reduced-search Dynamic Programming for Approximation of Polygonal Curves" Pattern Recognition Letters 24:pp2243-2254, 2003.

[8] Kolesnikov A., Fränti P., Wu X., "Multiresolution Polygonal Approximation of Digital Curves" Proceedings of the 17th International Conference on Pattern Recognition ICPR'04 pp855-858 2004.

[9] Marteau P.F., Ménier G. "Adaptive multiresolution and dedicated elastic matching in linear time complexity for time series data mining". In Sixth International Coference on Intelligent Systems Design and Applications, pp 700-706, 2006.

[10] Marteau, P.F., Time Warp Edit Distances with Stiffness Adjustment for Time Series Matching, IEEE Transactions on Pattern Analysis and Machine Intelligence, Volume: 31, Issue: 2, pp. 306-318, Feb. 2009.

[11] Melkman A., O'Rourke J. "On Polygonal chain approximation" Computationnal Morphology. Elsevier Science Publishers B.V. North Holland pp 87-95, 1988.

[12] Perez JC., Vidal E., "Optimum Polygonal Approximation of Digitized Curves" Pattern Recognition Letters 15:pp743 – 750, 1994.

[13] Pikaz A., Dinstein I.. "An algorithm for polygonal approximation based on iterative point elimination". Pat- tern Recognition Letters, 16:pp557-563, 1995.

[14] Rosin R.L. "Techniques for assessing polygonal approximations of curves". IEEE Trans. Pattern Analysis and Machine Intelligence, 14:pp659-666, 1997.

[15] Salotti M.. "An efficient algorithm for the optimal polygonal approximation of digitized curves". Pattern Recognition Letters, 22:pp215-221, 2001.

[16] Visvalingam M., Whyatt J. "Line generalization by repeated elimination of points". Cartographic Journal, 30:pp46-51, 1993

[17] Y. Zhu and L.D. Seneviratne. "Optimal polygonal approximation of digitized curves". In IEEE Proc.-Vis. Image Signal Process, volume 144, pp814-1997.