Phase Unwrapping and Operations Research

Thibaut Vidal

Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro
Rua Marquês de São Vicente, 225 - Gávea, Rio de Janeiro - RJ, 22451-900, Brazil
vidalt@inf.puc-rio.br

Seminar GERAD
Montreal, February 4th, 2016

Joint work with Ian Herszterg and Marcus Poggi

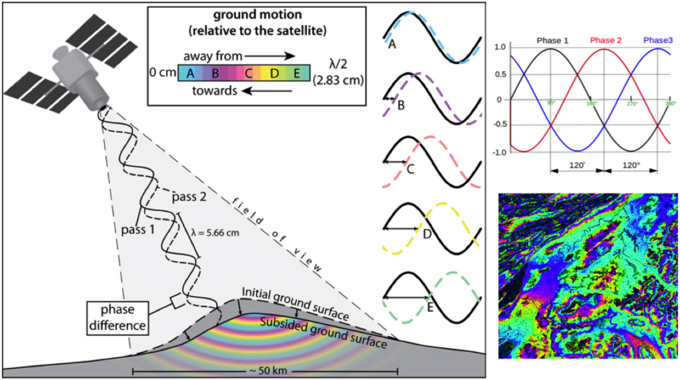


Figure : Radar interferometry

Wrapped phase

- While the phase information can take any real value, it is wrapped to a 2π interval with a $]-\pi, \pi]$ domain by the $\arctan$ operator.

Wrapped phase

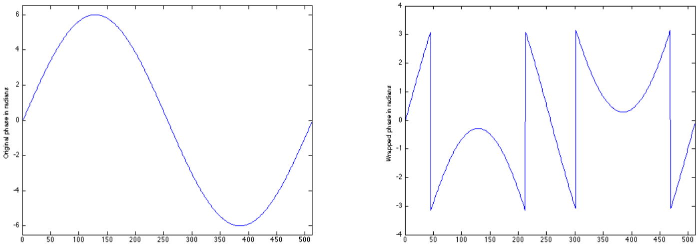


Figure : *Wrapping effect on a 1D continuous phase signal*

Wrapped phase

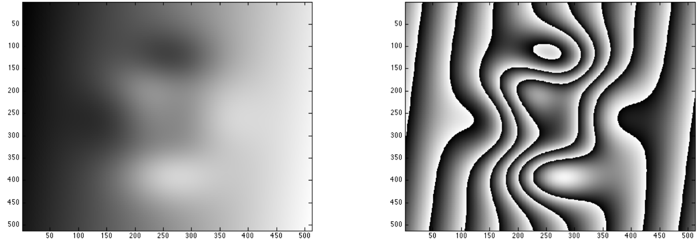


Figure : *Wrapping effect on a 2D phase image*

Phase Unwrapping

- **Phase Unwrapping** = reconstructing the continuous signal by removing the 2π-multiple ambiguity.
- ITOH, K., Analysis of the phase unwrapping algorithm, Applied Optics, v.21, n.14, p. 2470-2470, 1982

Itoh's Unwrapping Method (for discretized phase values):

Input = Wrapped phase values, $\psi(n)$

Output = Unwrapped phase values $\phi(n)$

Initialization: $\phi(1) = \psi(1)$;

For $i \leftarrow 2$ to N

$\Delta_\psi \leftarrow \psi(i) - \psi(i-1)$;
IF $\Delta_\psi \leq -\pi$

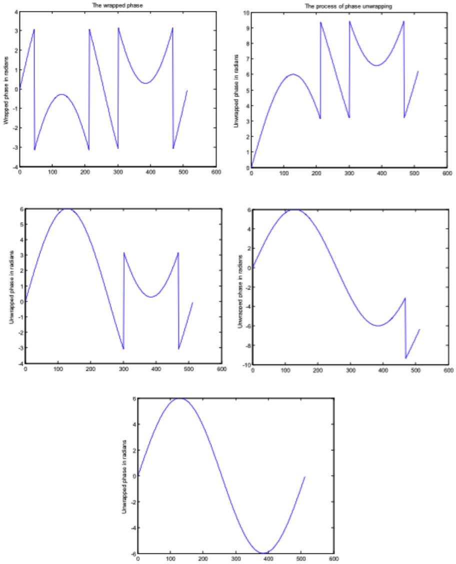
$\Delta_\psi \leftarrow \Delta_\psi + 2\pi$

ELSEIF $\Delta_\psi > \pi$

$\Delta_\psi \leftarrow \Delta_\psi - 2\pi$;

$\phi(i) \leftarrow \phi(i-1) + \Delta_\psi$;

Phase Unwrapping



Phase Unwrapping

Itoh's condition: For unambiguous phase unwrapping, the difference between any two adjacent samples in the continuous phase signal should not exceed a value of π

Phase Unwrapping

- Phase unwrapping problems often comes from complex applications dealing with rich geometries and signal acquisition methods that are highly susceptible to noise.
- Itoh's condition is not fulfilled $\Rightarrow$ Occurrence of *"fake wraps"*.
- Errors are propagated through subsequent samples in the unwrapping process.

Phase Unwrapping

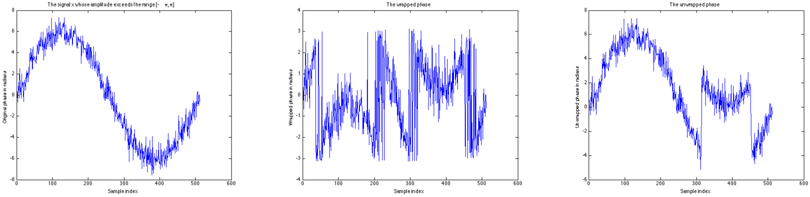


Figure : Unwrapping process over noisy data

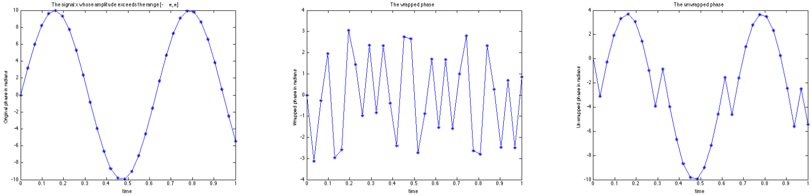


Figure : Unwrapping process over under-sampled data

Table of Contents

Residue theory

- **In higher dimension:** Itoh's algorithm can be applied to any continuous integration path

$\Rightarrow$ Every integration path P can constitute a discrete unwrapping path over any multidimensional space.

$\Rightarrow$ Paths could be selected to avoid damaged regions (noise, under-sampling)

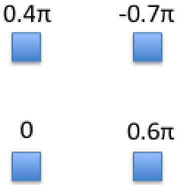
Residue theory

- How to detect singularities in two or more dimensions?

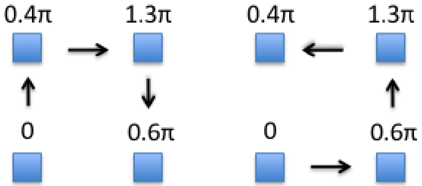
  GHIGLIA, D. C.; MASTIN, G. A.; ROMERO, L. A, Cellular automata method for phase unwrapping, v.4, n.1, p. 267-280, 1987

Residue theory

Wrapped phase example A – No singularity:

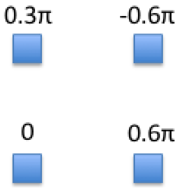


Unwrapped values from Example A:

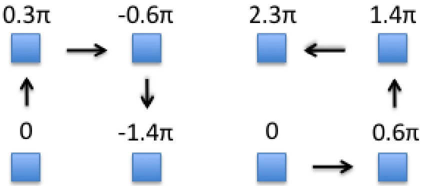


Residue theory

Wrapped phase example B – Singularity:



Unwrapped values from Example B:

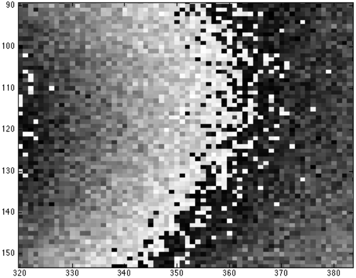


Residue theory

⇒ The location of all singularities can be identified by checking all 2x2 elementary loops (Ghiglia & Pritt, 1998). These specific points are called "residues"

- Residues charges (polarity) are either positive (+1) or negative (-1)
- In the presence of residues, an unambiguous phase unwrapping is possible if, and only if, every integration path encircles none or a *balanced* number of residues (as many positive as negative)

Residue theory



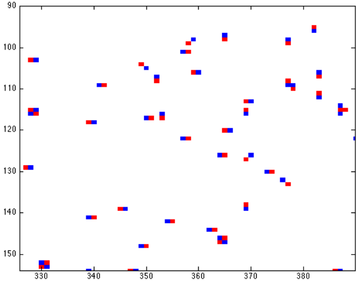


Figure : *Residues detected over a wrapped phase image corrupted by noise*

Residue theory

- Yet, not all residues come from noise
- Phase discontinuities are naturally present in many phase unwrapping applications.
- The topology of residues may suggest structural delimitations in the subject of study.

Residue theory

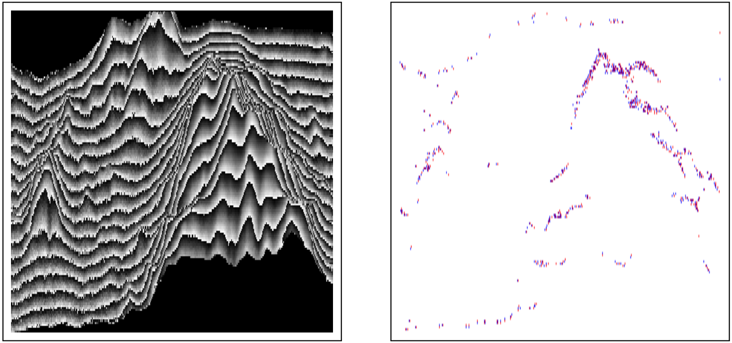
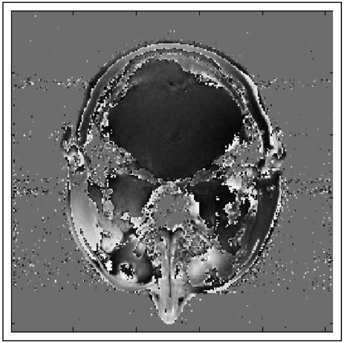


Figure : *Wrapped phase data and residues – high-fidelity InSAR simulator on a steep-relief mountainous region in Colorado*

Residue theory



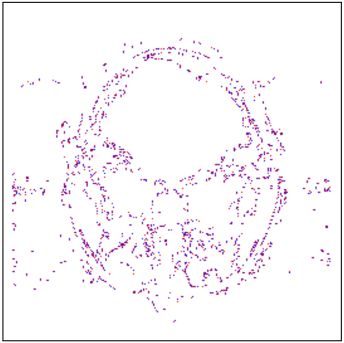


Figure : *Wrapped phase data and residues – head MRI*

Path-following Methods

- **Path-following Methods:**

⇒ Apply the path unwrapping method, but the solution is unique if and only if no integration path can encircle an unbalanced number of residues

⇒ For this purpose, create artificial barriers called *branch-cuts* to solve the path-dependency problem.

⇒ Branch-cuts can introduce a $\pm 2\pi$ discontinuity between samples in opposite sides of the barriers.

Path-following Methods

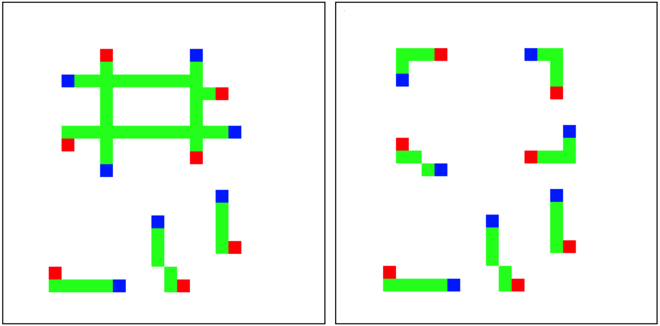


Figure : *Example of residues (blue, red) and possible branch-cuts configurations (green).*

Path-following Methods

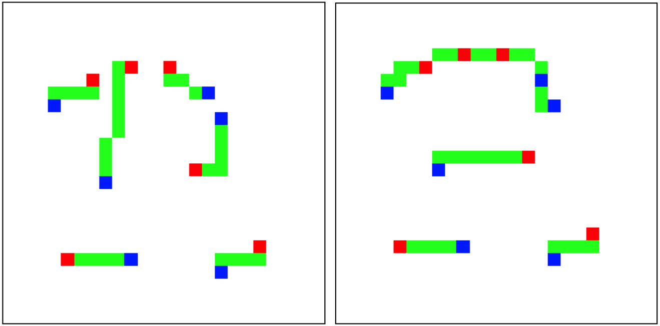


Figure : *Another example of residues (blue, red) and possible branch-cuts configurations (green).*

Path-following Methods

- The placement of branch-cuts fully characterizes the unwrapped solution

$\Rightarrow$ **Matching pairs of residues** is possible (Buckland et al. 1995. *Unwrapping noisy phase maps by use of a minimum-cost-matching algorithm*. Applied Optics, 44(0))

$\Rightarrow$ **Creating trees of residues** is possible, as long as they include a balanced number of positive and negative residues (or are connected to the border of the image).

$\Rightarrow$ **Using Steiner points** is possible

- Minimizing the length of the branch-cuts is a variant of geometrical Steiner problem with additional balance constraints $\Rightarrow$ NP-hard (and quite “tough” in practice for heuristic and exact methods)

Norm minimization

- Finally, from a "norm-minimization" perspective

⇒ Seeking a continuous solution whose gradients are "as close as possible" to those of the wrapped signal (norm minimization)

$$\arg\min_{\Phi} \sum_{m=1}^{M} \sum_{n=1}^{N-1} \left|\Delta^h \phi_{m,n} - \Delta^h \psi_{m,n}\right|^p + \sum_{m=1}^{M} \sum_{n=1}^{N-1} \left|\Delta^v \phi_{m,n} - \Delta^v \psi_{m,n}\right|^p$$

$$\textbf{\textit{s.t.}} \quad \phi_{m,n} = \psi_{m,n} + 2\mathbf{k}_{m,n}\pi \quad \forall (m,n)$$

$$\mathbf{k} \in \mathbb{Z} \quad \forall (m,n)$$

⇒ Remark that the length of the branch cuts is an upper bound of the number of differences of gradient (L^0-norm) between the wrapped signal and the continuous solution.

Table of Contents

Proposed methodology:

- We search for a minimum-cost balanced spanning forest (MCBSF)
- Spanning trees are allowed, as long as they contain a balanced number of residues, or are connected to the border of the image
- *We do not include Steiner points in the solutions.*

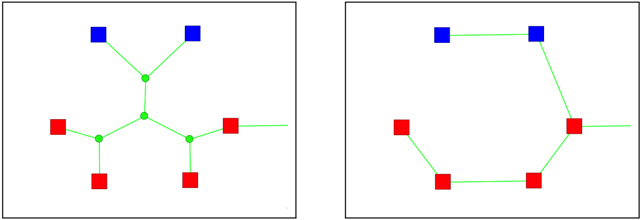


Figure : *Using Steiner-trees to cluster groups of residues.*

Main assumptions

- *We do not include Steiner points in the solutions* ⇒ **Why?**

⇒ Spanning trees better respect the natural boundaries of the image (cliffs, fractures...)

⇒ For most practical purposes, the optimal spanning tree solution is a high-quality approximation of the Steiner solution.

⇒ The model remains NP-hard, but efficient combinatorial optimization methods can be developed.

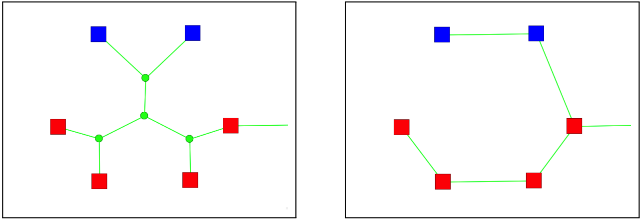


Figure : *Using Steiner-trees or spanning trees to cluster groups of residues.*

Mathematical formulation: Cut-based

Let $G = (V,E)$ be a graph with positive edge costs, where every vertex $v \in V$ has a weight $w_v \in \{-1, 1\}$. Let d_e be the cost (distance) of edge $e \in E$ and x_e be the decision variable indicating whether edge e should be part of the solution.

Mathematical formulation: Cut-based

$$
\begin{aligned}
\min \sum_{e \in E} d_e x_e & \\
\text{s.t.} \sum_{a \in \delta^+(S)} x_a \geq 1 & \qquad \forall S \subset V \text{ such that } \sum_{v \in S} w_v > 0 \\
\sum_{a \in \delta^-(S)} x_a \geq 1 & \qquad \forall S \subset V \text{ such that } \sum_{v \in S} w_v < 0 \\
x_e + x_{e'} \leq 1 & \qquad \forall e = (i,j), e' = (j,i) \in E \\
x_e \in \{0,1\} & \qquad \forall e \in E
\end{aligned}
$$

Mathematical formulation: Set Partitioning

- Set Partitioning formulation (SPF) for the MSFBC:
  - Let J be the set of all balanced subsets V_j of V
  - c_j is the cost of the MST connecting subset V_j

$$\min \sum_{j \in J} c_j x_j$$

$$\text{s.t.} \quad \sum_{j \in J} a_{vj} x_j = 1 \qquad \forall v \in V$$

$$x_j \in \{0,1\} \qquad \forall j \in J$$

$$a_{vj} = \begin{cases} 1 & \text{if } v \in S_j \\ 0 & \text{if } v \notin S_j \end{cases}$$

Solution Methods

- We developed mathematical programming approaches using the cut-based formulation, and metaheuristics
  - Primal Heuristics (metaheuristics)
  - Dual Heuristic + Dual Ascent to discard non-promising arcs
  - Branch-and-cut algorithm

Solving the linear program

- Because of the exponential number of constraints, unrealistic to solve even the linear program ⇒ cuts generation.

```
input  : The instance of the problem
output: The optimal solution set
Initialization: Solve the initial LP considering only the cuts with single
vertices as constraints. Let x be the solution set, lp the current linear
program and balanced a boolean indicating if all trees are balanced.
x ← solve(lp)
balanced ← false;
while not balanced do
    Build G̅ = (V̅, E̅) from the solution set x;
    balanced ← true;
    foreach pair (i,j) of vertices in V̅ do
        {S, maxFlow} ← minCutMaxFlow(G, i, j);
        if S is unbalanced, maxFlow < 1 and S ∉ lp then
            lp ← lp + {S};
            balanced ← false;
        end
    end
    if balanced is false then
        x ← solve(lp);
    end
end
return x;
```

Dual Heuristic

- Dual Heuristics: **Dual Ascent** (over the cut-based, directed, formulation)
  - ⇒ Selects violated cuts and increase their dual variables until one arc becomes saturated
  - ⇒ Based on Wong's Dual Ascent procedure for the Steiner-tree problem with directed cuts formulation (Wong, R.T., 1984. *A dual ascent approach for steiner tree problems on a directed graph*. Mathematical Programming, 28(3), pp.271-287.)
  - ⇒ Selection: Greedy or Random

Dual Ascent + Dual Scaling

- The selection of violated cuts was tested with two different criteria:
  - (**minrc**) by the minimum reduced cost arc in the graph (the cut that contains a minimum reduced cost-edge in its edge set)
  - (**random**) by randomly selecting a non maximal dual variable and saturating at least one of its arcs
- **Dual Scaling**
  - Multiplying the dual solution by a constant factor $0 < \alpha < 1$, and reapplying the dual ascent

Dual Heuristic

input : A dual initial solution π
output: A feasible dual solution π'

Initialization: *Build $G_\pi = (V, E)$ from the saturated arcs in π*
$\pi' \leftarrow \pi$;
while *exists a violated cut $R \in G_\pi$* **do**
    $W \leftarrow$ *selectViolatedCut()*;
    if $\sum_{v \in W} p_v > 0$ **then**
        Augment π'_W until at least one arc in δ^-(W) becomes saturated;
    end
    else if $\sum_{v \in W} p_v < 0$ **then**
        Augment π'_W until at least one arc in δ^+(W) becomes saturated;
    end
    Add the newly saturated arcs in G_π;
end
return π';

Branch-and-cut

- **Branch-and-Cut**
  - Based on the directed formulation
  - Uses primal bounds and dual bound to fix arcs by reduced cost
  - Uses the unbalanced cuts of the dual solution as initial constraints
  - Solves the linear relaxed program at each node
  - Branching: choose the most fractional variable
  - Exploration: depth-first search

Primal Heuristic – ILS

- Iterated Local Search (ILS) metaheuristic
  - Using an **indirect solution representation**: a solution is represented as a partition of the set of vertices into *components* $P_1, \ldots, P_k$ such that $\bigcup P_i = V$
  - The cost $c(P_i)$ of any component P_i can be efficiently derived by solving a minimum-cost spanning tree problem.
  - Any unbalanced component P_i is not considered infeasible, but must be connected to a dummy node that represents the border of the image.

- Iterated Local Search (ILS) metaheuristic
  - **Initial Solution** obtained by computing a minimum-cost spanning tree over V and disconnecting edges that are longer than a threshold d_{MAX}.
  - **Local Search** based on a variety of neighborhoods.
  - **Large neighborhood search** using mathematical programming over a set partitioning formulation
  - **Simple perturbation procedure**

Primal Heuristic – ILS

Algorithm 1 Hybrid Iterated Local Search

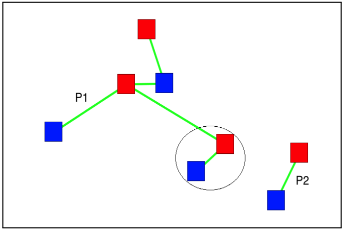
$S \leftarrow$ GenerateInitialSolution;
$S^* \leftarrow S$; $It_{\text{SHAK}} \leftarrow 0$;
while $It_{\text{SHAK}} < It_{\text{MAX}}$ **do**
    $S \leftarrow$ LocalSearch(S);
    if $\exists\, k \in \mathbb{N}^+$ s.t. $It_{\text{SHAK}} = k \times It_{\text{SP}}$ **then**
        $S \leftarrow$ SetPartitioning();
    end if
    if $c(S) < c(S^*)$ **then**
        $S^* \leftarrow S$;
        $It_{\text{SHAK}} \leftarrow 0$;
    end if
    $S \leftarrow$ Perturb(S) or Perturb(S^*) with equal probability;
end while
return S^*;

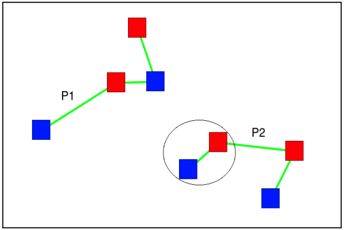
Primal Heuristic – ILS

- Iterated Local Search (ILS) metaheuristic
    - **Local Search** based on a variety of neighborhoods.
    - Enumerating all component pairs (T_i, T_j) in random order to test the associated moves.
    - Each move evaluation requires to build the new spanning trees for the modified components
    - First-improvement policy

ILS – Neighbourhoods

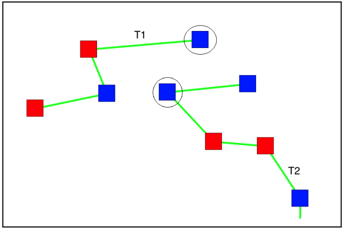
- Relocate - single vertices +/- or pairs (+,-)
  - Relocates one or more vertices v_i from T_1 to T_2, independently of its polarity, or any pair of opposite signed vertices v_i and v_j from T_1 to T_2.

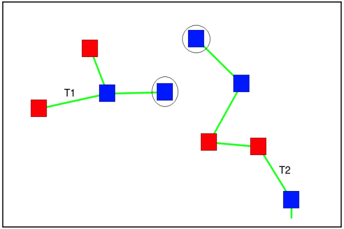




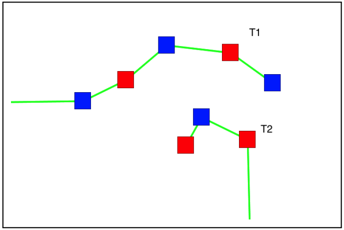
ILS – Neighbourhoods

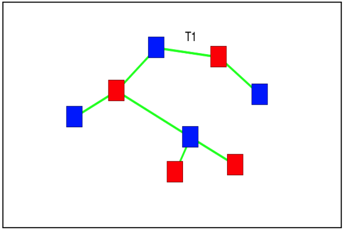
- Swap - single vertices +/- or pairs (+,-)
  - Swaps one or more vertices v_i from T_1 with v_j from T_2, both with same polarity, or a pair of vertices v_i and v_j from T_1 and a pair v_k and v_l from T_2 with opposite signed polarities.





- Merge
  - Merges two components T_1 and T_2, into a single component

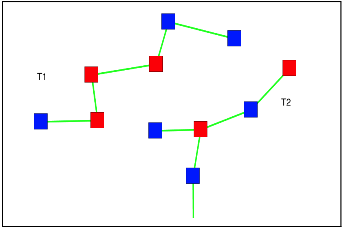


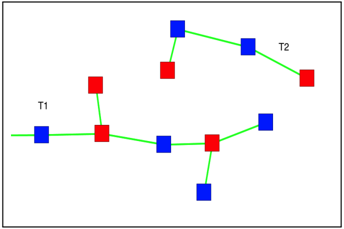


- Break
  - Breaks the longest edge in a given tree T_2, generating two new components

ILS – Neighbourhoods

- Break1-Insert1
  - Merges two given trees, T_1 and T_2, into a single component, compute the spanning tree and disconnect the longest edge, forming two new components.





ILS – Local Search

- **Speed-up** procedures:
  - **Memory structures** to avoid testing again moves that are known to be non-improving.
  - **Pruning**: avoids moves on trees that are very distant from each other by computing a maximum distance radius for each vertex.

ILS – Perturbation

- **Perturbation procedure** is applied to escape from local minima.
- Applied with equal probability to either S or S^*
- From the spanning-tree representation of the solution, with T, components, the perturbation removes $k \in \{1, \lceil 0.15T \rceil\}$ edges, creating disjoint components which are randomly recombined to resume the search with T components.

ILS – Set Partitioning

- Regularly solving the set partitioning formulation, using a pool of columns collected from local minimums of the ILS.
- The size of the pool is limited to 2000 columns.
- Executed every It_{SP} iterations.

Table of Contents

Overview

- Computational experiments designed to address two main objectives:
  - Validade and investigate the performance of the proposed methods.
  - Evaluate the performance of the MSFBC approach in the two-dimensional phase unwrapping domain, when compared to other path-following methods.
- Instances designed to test and identify the limitations and the scalability factor of the proposed methods.

Benchmark Instances

- Generated by randomly spreading p positive and n negative vertices on a $4p \times 4n$ Euclidean space.
- Complete graph: edge costs defined by the 2D Euclidean distance between vertices
- Every vertex is also connected to its closest border point
- 21 sets of 5 instances each: 8 to 1024 nodes
- We have collected the best solutions ever found during the heuristics and exact methods in order to evaluate the quality of each proposed algorithm.

Experiments – Hybrid ILS

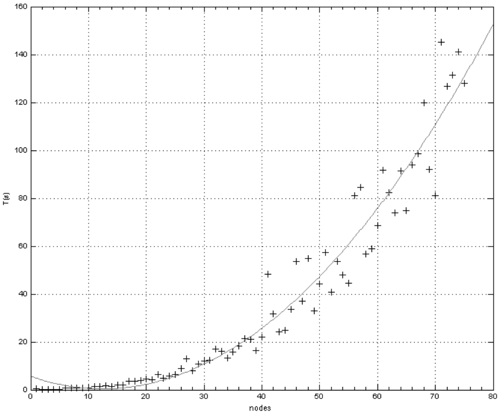
- Executed for 10 times with two termination criterion per run, whichever came first:
  - 100 iterations ($It_{MAX} = 100$) without improving the best solution found
  - A time bound of 3600 seconds
- The set covering routine is executed at every $(1/3)It_{MAX}$ iterations, with a time bound of 300 seconds
- The maximum distance radius for every vertex v is limited to 25% of the shortest distances between v and the set of vertices V-$\{v\}$

Experiments – Hybrid ILS

Group	$\|V\|$	GAP$_B$(%)	OPT	GAP$_{AVG}$(%)	Avg_T
PUC-8	8	0.00	**5/5**	2.49	0.27
PUC-12	12	0.00	**5/5**	0.00	0.77
PUC-16	16	0.00	**5/5**	0.52	1.60
PUC-20	20	0.00	**5/5**	0.21	3.60
PUC-24	24	0.00	**5/5**	0.64	5.46
PUC-28	28	0.00	**5/5**	1.06	10.50
PUC-32	32	0.00	**5/5**	0.76	14.88
PUC-36	36	0.00	**5/5**	0.94	19.91
PUC-40	40	0.00	**5/5**	0.63	32.49
PUC-44	44	0.14	**4/5**	1.44	44.50
PUC-48	48	0.00	**5/5**	0.73	48.91
PUC-52	52	0.00	**5/5**	1.33	70.03
PUC-56	56	0.00	**5/5**	1.35	82.76
PUC-60	60	0.00	**5/5**	1.15	97.13
PUC-64	64	0.35	4/5	3.02	134.45
PUC-80	80	0.40	3/5	3.26	304.64
PUC-96	96	0.10	**2/2**	4.81	650.05
PUC-128	128	1.05	**2/2**	5.45	2091.65
PUC-256	256	0.00	0/0	5.78	3600.00
PUC-512	512	0.00	0/0	6.31	3600.00
PUC-1024	1024	4.00	0/0	4.85	3600.00

Experiments – Hybrid ILS

- Growth of the CPU time appears to be cubic as a function of instance size.



Dual Ascent + Dual Scaling

Group	$\text{GAP}_{minrc}(\%)$	$\text{GAP}_{random}(\%)$	$\text{T}_{minrc}(\text{s})$	$\text{T}_{random}(\text{s})$
PUC-8	2.09	0.00	<0.01	<0.01
PUC-12	7.33	0.00	<0.01	<0.01
PUC-16	2.75	3.55	<0.01	<0.01
PUC-20	6.46	1.45	<0.01	<0.01
PUC-24	3.78	3.78	<0.01	<0.01
PUC-28	8.92	3.34	0.01	<0.01
PUC-32	9.19	3.71	0.01	<0.01
PUC-36	14.63	7.61	0.01	<0.01
PUC-40	13.05	2.96	0.02	0.01
PUC-44	13.82	3.74	0.02	0.01
PUC-48	4.78	3.22	0.02	0.01
PUC-52	11.93	4.21	0.03	0.02
PUC-56	10.80	3.29	0.03	0.02
PUC-60	10.32	3.99	0.07	0.03
PUC-64	11.14	4.15	0.08	0.03
PUC-80	15.94	6.87	0.13	0.07
PUC-96	18.69	9.19	0.11	0.12
PUC-128	16.37	10.28	0.36	0.30
PUC-256	34.81	16.48	10.48	2.64
PUC-512	33.87	18.18	53.37	20.02
PUC-1024	40.44	26.15	1312.55	169.29

Branch-and-cut

- Executed with a time bound of 3600 seconds
- 80 out of 105 primal solutions obtained by the ILS method proved to be optimal
- 20 out of 105 instances were not solved to optimality, with an average gap of 17% between the best lower and upper bounds
- As expected, the separation of cuts by the min-cut/max-flow procedure took more than 50% of the running time in many instances

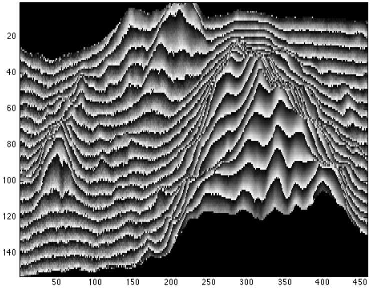
Branch-and-cut

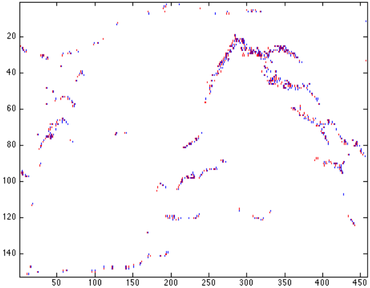
Group	V	Reduction (%)	GAP_{root}(%)	GAP_{LB-UB}(%)	OPT	Avg-T (s)	PtPS(%)
PUC-8	8	85.28	0.00	0.00	**5**	<0.01	4.47
PUC-12	12	92.05	0.00	0.00	**5**	<0.01	0.79
PUC-16	16	91.47	0.07	0.00	**5**	<0.01	9.64
PUC-20	20	93.95	0.42	0.00	**5**	0.01	8.98
PUC-24	24	85.63	0.04	0.00	**5**	0.01	9.05
PUC-28	28	86.95	1.36	0.00	**5**	0.03	11.98
PUC-32	32	85.93	1.54	0.00	**5**	1.13	14.52
PUC-36	36	66.95	2.32	0.00	**5**	2.27	32.37
PUC-40	40	79.10	1.33	0.00	**5**	1.15	52.03
PUC-44	44	84.56	1.75	0.00	**5**	9.43	43.30
PUC-48	48	88.39	0.82	0.00	**5**	0.39	35.11
PUC-52	52	85.93	1.34	0.00	**5**	1.76	32.35
PUC-56	56	74.77	1.17	0.00	**5**	30.61	37.13
PUC-60	60	72.58	1.45	0.00	**5**	4.37	39.82
PUC-64	64	74.62	1.46	0.00	**5**	128.94	55.28
PUC-80	80	48.02	2.05	0.00	**5**	1057.56	38.61
PUC-96	96	48.01	1.96	0.60	3	2182.13	54.17
PUC-128	128	55.32	2.95	2.62	2	2302.03	45.18
PUC-256	256	2.69	9.87	2.13	0	3600.00	34.03
PUC-512	512	0.00	17.47	15.34	0	3600.00	19.04
PUC-1024	1024	0.00	18.62	18.54	0	3600.00	15.56

2DPU: Instances & Metrics

- Methods tested with three well-known benchmark instances and compared against two classic 2DPU algorithms
- Four metrics in order to evaluate and compare the quality of each solution :
  - (**N**) the total number of absolute phase gradients that differ from their wrapped counterparts
  - (**L**) The total length of the branch-cuts
  - (**T**) The number of trees produced by the branch-cuts.
  - (**I**) The number of isolated regions produced by the branch-cuts

Long's Peak

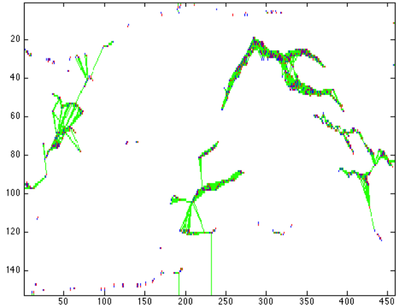

20
40
60
80
100
120
140
50
100
150
200
250
300
350
400
450

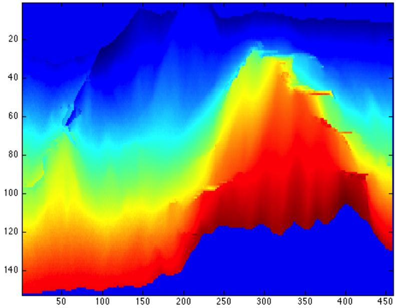

20
40
60
80
100
120
140
50
100
150
200
250
300
350
400
450

Long's Peak

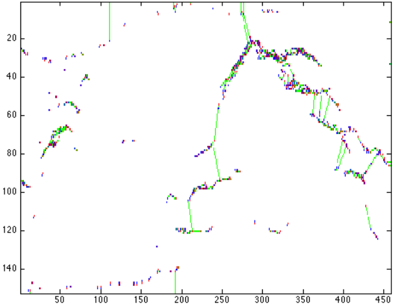
- Radar interferometry example
- 846 residues (422 positives and 426 negatives) distributed over a 152x458-pixel image
- Greatest challenge: Efficiently cluster the sparse group of residues and respect the structural delimitations

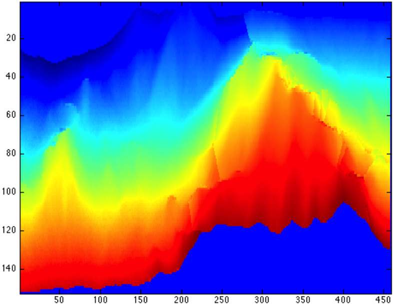
Long's Peak: Goldstein



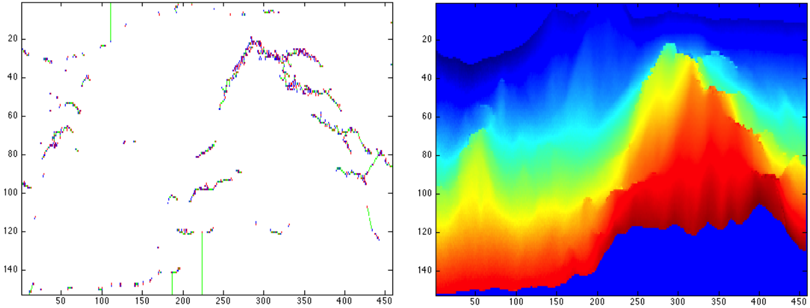


Long's Peak: Minimum-cost matching algorithm





Long's Peak: MSFBC

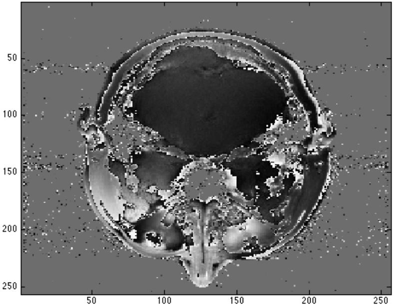


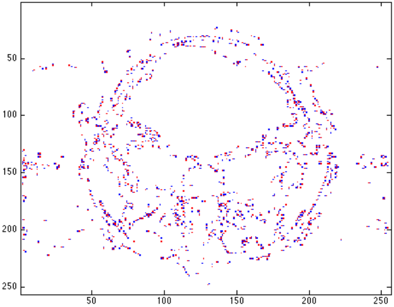
Long's Peak: Summary

Table : Results for Long's Peak data set

Method	N	L	T	I
Goldstein	1437	10647.96	49	110
MCM	1075	1545.38	429	47
MSFBCP	975	1264.31	68	25

Head Magnetic Resonance Image (MRI)

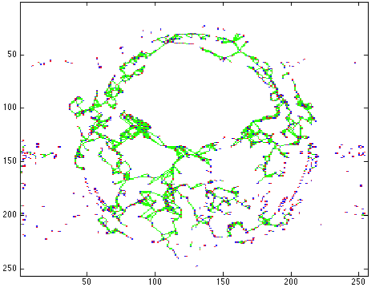


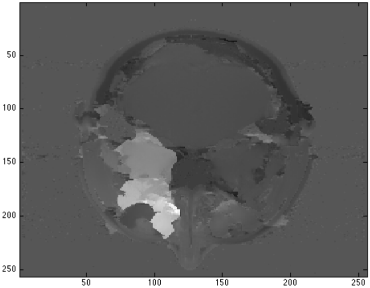


Head Magnetic Resonance Image (MRI)

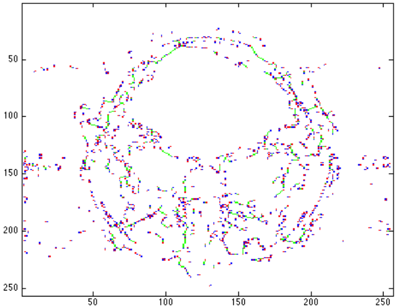
- Magnetic Resonance Image example
- 1926 residues (963 positives and 963 negatives) defined on a 256x256-pixel grid
- Greatest challenge: Considered to pose a difficult problem to the unwrapping procedure since various regions are delimited by residues and appear to be completely isolated from one another.

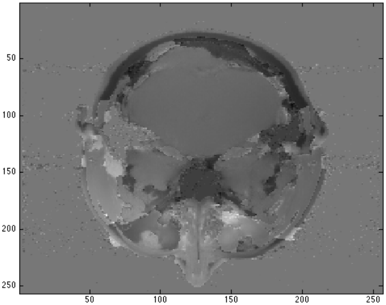
Head MRI: Goldstein



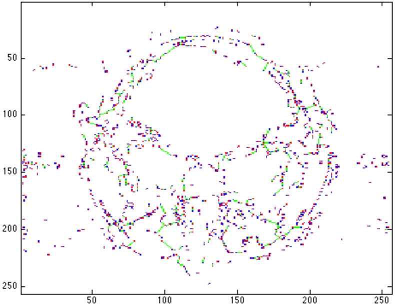


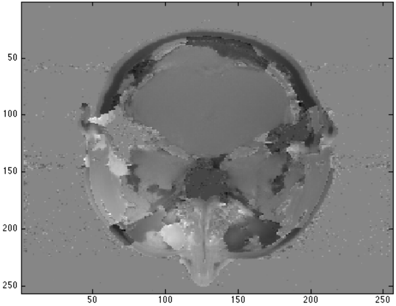
Head MRI: Minimum-cost matching algorithm





Head MRI: MSFBC





Head MRI: Summary

Table : Results for Head MRI data set

Method	N	L	T	I
Goldstein	2570	11696.44	153	257
MCM	1789	1588.72	963	16
MSFBCP	1810	1722.56	57	19

Table of Contents

Contributions

- We have proposed a new model for the 2D Phase Unwrapping problem, along with a new set of mathematical formulations and methods
- We developed efficient methods known from the field of optimization and operations research to address the minimization of the branch-cuts
- The proposed methods constituted a better approximation of the "L^0-norm" problem in the field of phase unwrapping

Discussions and Future work

- Solutions obtained by heuristic methods, with no guarantee on optimality
- In fact, the optimal solution for the MSFBC approach would be theoretically better than any path-following method
- Steiner $\times$ MSFBC?
- Devise a column generation approach and general improvements over the heuristic methods

Thanks!