2DPhase Unwrapping Problem (2DPU) via Minimum Spanning Forest with Balance Constraints

Ian Herzterg, **Marcus Poggi**, Thibault Vidal
Departamento de Informática, PUC-Rio, Brazil
{iherzterg,poggi, vidalt} @inf.puc-rio.br

June 16, 2015

2D-Phase Unwrapping

- In order to illustrate the importance of phase in signal processing, Oppenheim and Lim proposed an experiment:
  - Through the 2D discrete Fourrier transform (DFT), decompose an image into its sine and cosine component
  - This separates the complex spectrum into a magnitude and a phase value for each pixel.
  - The output of the transformation represents the image in the frequency domain, while the input image is the spatial domain equivalent.
  - In the Fourrier domain image, each pixel represents a particular frequency contained in the spatial domain.
  - This technique is used in a wide range of applications: image analysis, image filtering and image reconstruction

2D-Phase Unwrapping

- This process induces an inherent difficulty:
  - Since the phase ranges within 0 and $2.\pi$ a discontinuity arises every time it passes through this point, wrapping the image.
    .
  - ITOH, K., Analysis of the phase unwrapping algorithm, Applied Optics, v.21, n.14, p. 2470-2470, 1982
    .
    Showed that for an efficient phase unwrapping, any two adjacent samples in the continuous phase absolute phase difference signal cannot exceed the value of π

Itoh condition: $|\Delta_{\phi_n}| \leq \pi$

- The linear difference between adjacent samples can be defined as:

$$\Delta_{\phi_n} = \phi_n - \phi_{n-1}, \tag{1}$$

Thus

$$\sum_{n=1}^{m} \Delta\phi_n = \phi_m - \phi_0, \tag{2}$$

Leading to:

$$\Delta W(\phi_n) = (\phi_n + 2\pi k_n) - (\phi_{n-1} + 2\pi k_{n-1}) \tag{3}$$

$$\Delta W(\phi_n) = \phi_n - \phi_{n-1} - 2\pi(k_n - k_{n-1}). \tag{4}$$

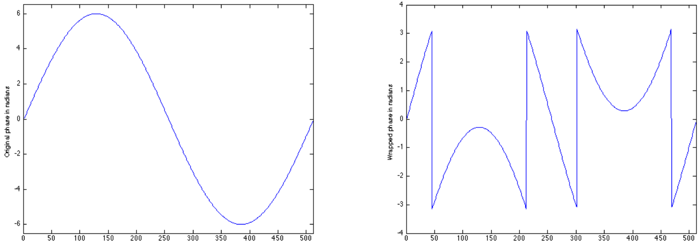


Figure: *Wrapping effect on a 1D continuous phase signal.*
(a) Continuous signal
(b) Wrapped signal.

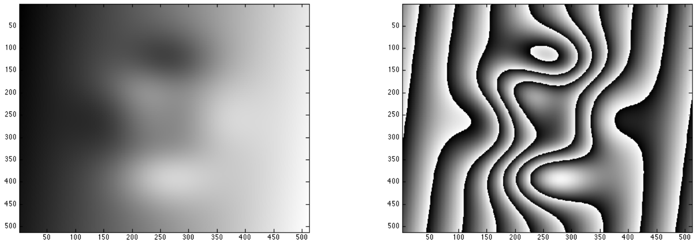


Figure: *Wrapping effect on a 2D phase image.*
(a) Absolute phase image
(b) Wrapped phase image.

Itoh's Unwrapping Method for Discretized Phase:

InputWrapped phase values, $\psi(n)$

OutputUnwrapped 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 $\Delta_\psi > \pi$
$\Delta_\psi \leftarrow \Delta_\psi - 2\pi$;
$\phi(i) \leftarrow \phi(i-1) + \Delta_\psi$;

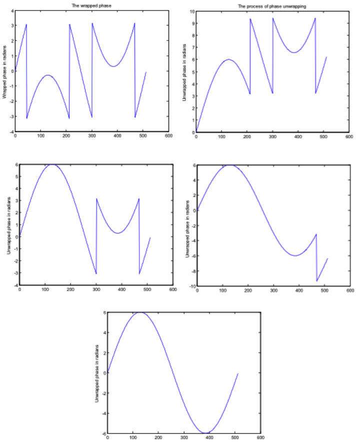


Figure: *Unwrapping process by the Itoh's method for 1D Phase Unwrapping, returning the wrapped phase signal to its original continuous form.*

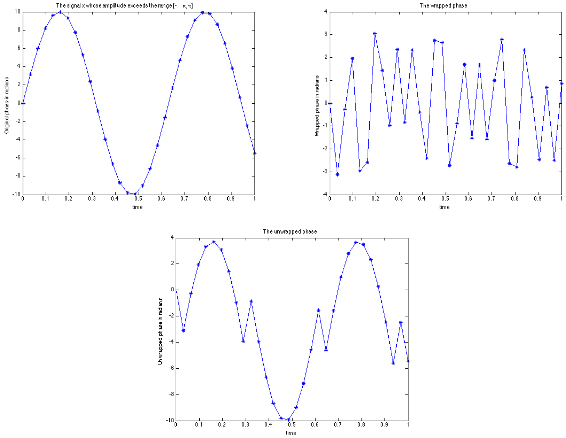


Figure: Unwrapping process over under-sampled data
(a) The continuous phase signal under-sampled.
(b) The wrapped under-sampled phase signal.
(c) The unwrapping obtained by the Itoh algorithm.

- **Itoh's can be applied to any continuous integration path**

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

  In more than 1D integration paths can be selected

  Paths are select to avoid damaged regions (noise, under-sampling)

- **Itoh's can be applied to any continuous integration path**

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

  In more than 1D integration paths can be selected

  Paths are select to avoid damaged regions (noise, under-sampling)

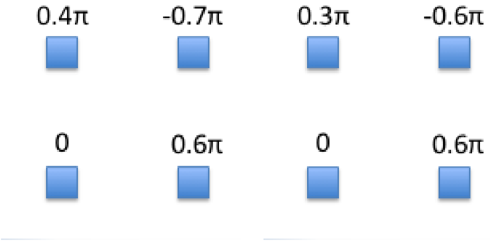


Figure: *2D Wrapped phase data examples.*
(a) Example A with no singularities present.
(b) Example B under-sampled.

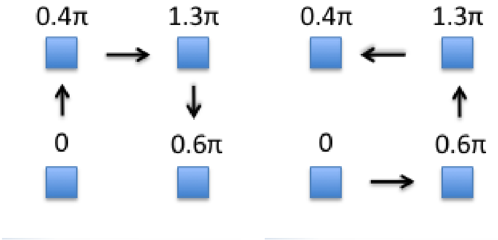


Figure: *Unwrapped values from Example A showing no path dependency. (a) Unwrapped values obtained by the clockwise integration path. (b) Unwrapped values obtained by the counterclockwise integration path.*

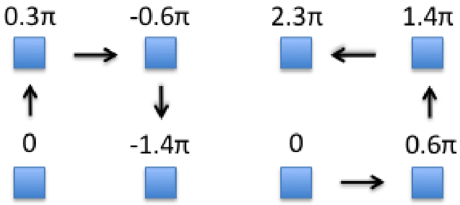


Figure: *Unwrapped values from Example B, showing the occurrence of path dependency.*
(a) Unwrapped values obtained by the clockwise integration path.
(b) Unwrapped values obtained by the counterclockwise integration path.

- The location of all residues can be identified by checking all 2x2 elementary loops (Ghiglia & Pritt, 1998)
- Residues charges (polarity) are either positive (+1) or negative (-1)
- When residues are present unwrapping is possible if, and only if, every integration path encircles none or a balanced number of residue charges.

GOLDSTEIN, R. M.; ZEBKER, H. A. ; WERNER, C. L. Satellite radar interferometry: Two-dimensional phase unwrapping, Radio science, v.23, n.4, p. 713-720, 1988.

- Path-following algorithms
  - Directly applies the line integration schemes
  - Assumes Itoh condition must hold along every integration path
  - Whenever this condition is not met, different integration paths may lead to different unwrapped solutions: path dependency

- Minimum norm methods
  - Based on the idea that the difference between absolute phase values among neighbour samples are equal to the wrapped differences of their correspondent wrapped phase values.
  - Finds a phase solution for which the L^p norm of the gap between these differences is minimized.

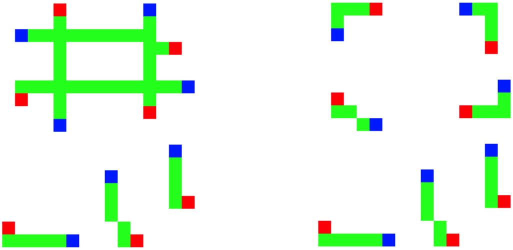


Figure: *Example of residues and possible branch-cuts configurations. (a) First branch-cut configuration, creating an isolated region. (b) Minimum length branch-cuts configuration.*

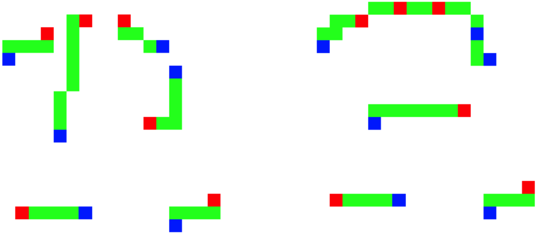


Figure: *Another example of residues and possible branch-cuts configurations.*
(a) Minimum length pair connections between positive and negative residues.
(b) Minimum length pair connection between balanced components.

- Goldstein et al.: Classic path-following algorithm
    - Effective at generating short length branch-cuts in an extremely fast way
    - Main idea: connect nearby residues with branch cuts until every component of connected residues becomes balanced
    - The cuts are generated by approximatively minimising the sum of the cut lengths

- Minimum-cost matching algorithms
- BUCKLAND, J.; HUNTLEY, J. ; TURNER, S., Unwrapping noisy phase maps by use of a minimum-cost-matching algorithm, Applied Optics, v.34, n.23, p. 5100-5108, 1995
  - Minimum-cost matching algorithm: solves the path-dependency problem by creating branch-cuts between close pairs of positive and negative residues
  - Treats the minimization of the branchcuts lengths as a global optimization problem
  - Models the problem as a minimum cost matching between vertices in a bipartite graph.

- Minimum Spanning Forest with Balance Constraints (MSFBC)
  - Part of the family of path-following methods
  - Goal: Find an optimal branch-cut configuration which eliminates the path dependency problem
  - Seeks for a minimum cost spanning forest which balances positive and negative residues in each tree
  - A tree is balanced when connects an equal number of positive and negative residues or it contains a border point in its vertex set

- Problem formulation
  - Let $G = (V,E)$ be the graph where the vertex set $V = (R \cup B)$ represents the union between the sets of residues R and border points B, where $p_r \in$ {-1,1} denoting the polarity for every residue $r \in R$, $B = \{b_r$, for every $r \in R\}$, $E = (E_R \cup E_B)$ where $E_R = \{(i,j)$,
  - $i,j \in R$, $i \neq j\}$ and $E_B = \{(r,b_r), r \in R, b_r \in B\}$, d_e is the cost (distance) of edge $e \in E$ and x_e is the decision variable indicating whether edge e should be part of the solution.

- The undirected formulation for the MSFBC:

$$\min \sum_{e \in E} d_e x_e \quad s.t. \tag{5}$$

$$\sum_{e \in \delta(S)} x_e \geq 1, \quad \forall S \subset V, \quad s.t. \; \sum_{v \in S} p_v \neq 0 \tag{6}$$

$$x_e \in \{0, 1\}, \forall e \in E, \tag{7}$$

- The directed formulation for the MSFBC:

$$\min \sum_{e\in E} d_e x_e \quad \boldsymbol{s.t.} \tag{8}$$

$$\sum_{a\in\delta^+(S)} x_a \geq 1, \quad \forall S \subset R, \quad s.t. \quad \sum_{v\in S} p_v > 0 \tag{9}$$

$$\sum_{a\in\delta^-(S)} x_a \geq 1, \quad \forall S \subset R, \quad s.t. \quad \sum_{v\in S} p_v < 0 \tag{10}$$

$$x_a + x_{a'} \leq 1, \quad \forall a = (i,j), a' = (j,i) \in E_R \tag{11}$$

$$x_a \in \{0,1\}, \forall a \in E \tag{12}$$

Solving the Minimum Spanning Forest with Balance Constraints (MSFBC)

- Primal Heuristics - Iterated Local Search (ILS)
  Tree Operations
  - Relocate (residue), Relocate-C (pairs of residues)
  - Swap (residue), Swap-C (pairs of residues)
  - Merge, Break (connects of disconnect 1 edge)
  - Break-1-Insert-1: Computes MST over all vertices of two trees then breaks removing the largest edge
- Dual Heuristics
  - Dual Ascent: Selects connected components until they become balanced
  - Selection: Greedy and Random
  - Reverse delete step to generate primal solution

- Solving the Minimum Spanning Forest with Balance Constraints (MSFBC)
  - Exact Algorithm
    - Branch-and-cut over the directed formulation
    - Uses primal bounds and dual bound to fix by reduced cost
  - Linear Relaxation
    - Accelerates cut separation by testing whether connect components considering all edges with positive value are balanced
    - Solves Max-Cut for all pairs

- Does a solution with a smaller value for the MSFBC implies a better unwrapping ?
  - This means that it is possible to connect the residues in order to connect them with

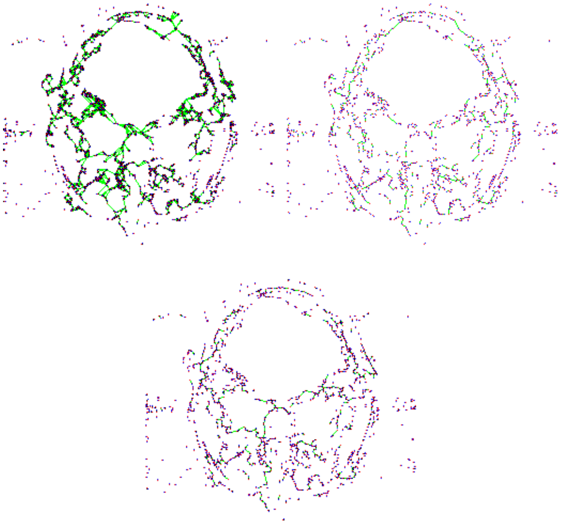


Figure: *Head. (a) Goldstein (b) Matching (c) MSFBC*

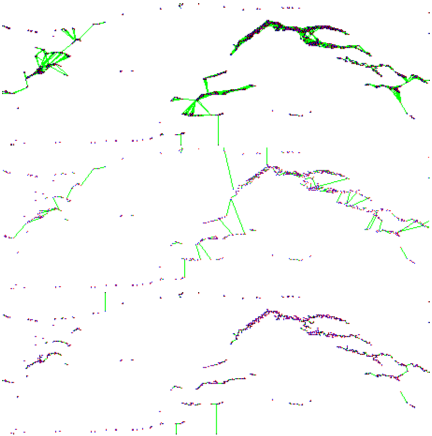


Figure: *Long (a) Goldstein (b) Matching (c) MSFBC*

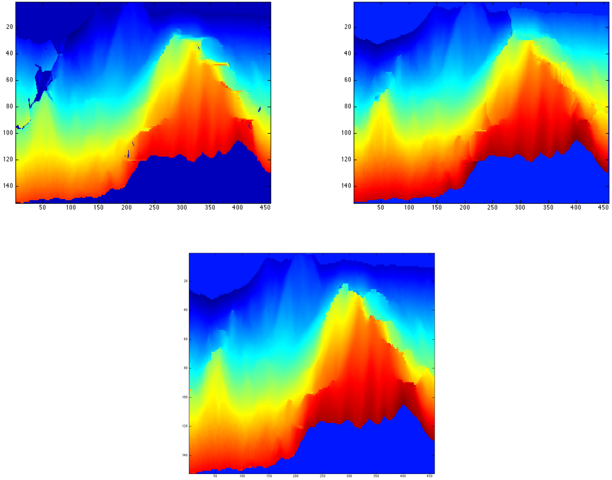


Figure: *Long (a) Goldstein* *(b) Matching* *(c) MSFBC*

Instances

- MSFBC - How good are we ?
  - Preliminary Tests on Random Instances
    - Equal number of positive and negative residues
    - Report on instance from 20 pairs to 128 pairs (40 to 256 vertices)

Primal Heuristics Results

Instance	|V|	|E|	Best Known	Best Sol.	GAP (%)	Avg Sol.	GAP (%)	T(s)
PUC_p_20_20_1	40	1640	347.490987	347.490984	0	354.319823	1.92730848	22.881383
PUC_p_20_20_2	40	1640	381.249147	381.249146	0	381.327912	0.02065545	32.124351
PUC_p_20_20_3	40	1640	391.298766	391.298766	0	398.147053	1.72003961	23.350068
PUC_p_20_20_4	40	1640	399.452595	405.228056	1.4252372	411.155273	2.8462916	24.614464
PUC_p_20_20_5	40	1640	366.812088	366.812088	0	374.605646	2.08046998	28.361182
PUC_p_22_22_1	44	1980	459.009082	462.137305	0.67690337	469.388708	2.21130714	33.775487
PUC_p_22_22_2	44	1980	413.546758	413.546758	0	420.068548	1.55255375	34.408586
PUC_p_22_22_3	44	1980	433.645458	442.351592	1.968148	451.767502	4.01136512	34.438296
PUC_p_22_22_4	44	1980	461.944699	467.667941	1.22378327	473.884787	2.51961834	36.658451
PUC_p_22_22_5	44	1980	448.584133	452.433285	0.85076676	462.129164	2.93100545	27.99622
PUC_p_24_24_1	48	2352	460.297172	473.141817	2.71475582	477.858422	3.67499016	54.373147
PUC_p_24_24_2	48	2352	477.413172	478.535943	0.23462626	489.123224	2.39409037	37.952988
PUC_p_24_24_3	48	2352	418.466013	420.569845	0.50023368	428.616298	2.3681519	51.868471
PUC_p_24_24_4	48	2352	459.073238	459.073238	0	467.90797	1.88813454	44.235078
PUC_p_24_24_5	48	2352	480.622508	480.622509	0	482.617076	0.41328169	46.238577
PUC_p_26_26_1	52	2756	573.995609	582.537393	1.46630656	595.551952	3.61955711	50.434437
PUC_p_26_26_2	52	2756	515.261508	516.54997	0.24943608	544.385048	5.34980527	46.0252
PUC_p_26_26_3	52	2756	594.352813	594.352808	0	605.37717	1.82107247	50.50561
PUC_p_26_26_4	52	2756	472.589049	472.589049	0	481.825064	1.91688139	48.772453
PUC_p_26_26_5	52	2756	585.795691	585.795691	0	599.851119	2.34315275	47.090796

Primal Heuristics Results

Instance	|V|	|E|	Best Known	Best Sol.	GAP (%)	Avg Sol.	GAP (%)	T(s)
PUC_p_28_28_1	56	3192	640.926754	640.926754	0	660.130438	2.90907416	51.035219
PUC_p_28_28_2	56	3192	631.736623	653.340619	3.30669721	658.222369	4.02382952	60.610623
PUC_p_28_28_3	56	3192	594.231017	594.356366	0.02108987	602.644501	1.39609405	61.931711
PUC_p_28_28_4	56	3192	570.737114	580.508368	1.68322363	597.133754	4.42055734	53.687048
PUC_p_28_28_5	56	3192	544.909304	556.618073	2.1035553	580.436666	6.12079906	58.605564
PUC_p_30_30_1	60	3660	639.400587	651.588592	1.8705062	670.847858	4.68769045	72.66549
PUC_p_30_30_2	60	3660	722.46814	722.46814	0	747.89598	3.39991666	66.700597
PUC_p_30_30_3	60	3660	732.901971	753.166973	2.69063869	777.636143	5.75258396	61.960019
PUC_p_30_30_4	60	3660	668.176955	678.266327	1.48752365	699.75634	4.51291159	73.390249
PUC_p_30_30_5	60	3660	706.649792	725.858259	2.64631101	745.847322	5.25543618	71.014367
PUC_p_32_32_1	64	4160	788.831899	819.672017	3.76249492	836.865747	5.73973163	82.137952
PUC_p_32_32_2	64	4160	748.655533	748.655533	0	775.444663	3.45467978	86.014673
PUC_p_32_32_3	64	4160	754.848532	772.585634	2.29581049	794.156691	4.94967296	83.272617
PUC_p_32_32_4	64	4160	835.28175	810.715764*	-3.0301601	814.770556	-2.5174196	77.301989
PUC_p_32_32_5	64	4160	789.848633	819.682852	3.63972711	849.671973	7.04075713	68.608809

Primal Heuristics Results

Instance	|V|	|E|	Best Known	Best Sol.	GAP (%)	Avg Sol.	GAP (%)	T(s)
PUC_p_40_40_1	80	6480	1140.72296	1175.02735	2.91945452	1240.54077	8.04631458	139.668138
PUC_p_40_40_2	80	6480	1211.5083	1149.242523	-5.4179841	1156.5979	-4.7475795	140.176245
PUC_p_40_40_3	80	6480	1243.62585	1179.580984	-5.429459	1186.28108	-4.8339952	144.283866
PUC_p_40_40_4	80	6480	1158.59005	1202.97873	3.6898975	1228.6347	5.70101529	133.927819
PUC_p_40_40_5	80	6480	1122.35471	1244.5755	9.82027912	1216.26022	7.72083982	134.764142
PUC_p_48_48_1	96	9312	1373.49672	1436.65399	4.39613649	1476.01502	6.94561369	209.85237
PUC_p_48_48_2	96	9312	1614.87205	1680.82656	3.92393305	1724.53571	6.35902523	196.932821
PUC_p_48_48_3	96	9312	1727.82459	1555.63628	-11.068674	1600.96529	-7.9239251	202.936085
PUC_p_48_48_4	96	9312	1701.77234	1564.419607	-8.7797885	1602.35908	-6.2041814	197.232113
PUC_p_48_48_5	96	9312	1422.95151	1498.13847	5.01869216	1551.57455	8.28983938	221.069486
PUC_p_64_64_1	128	16512	2724.85596	2602.285193	-4.7101203	2649.66695	-2.8376777	433.657401
PUC_p_64_64_2	128	16512	2195.65564	1567.213802	-40.099305	2365.56494	7.18260976	450.804949
PUC_p_64_64_3	128	16512	2536.14063	2576.42139	1.56343843	2642.63719	4.029935	407.713675
PUC_p_64_64_4	128	16512	2546.45801	2461.563244	-3.4488151	2519.28245	-1.0787024	435.588076
PUC_p_64_64_5	128	16512	2269.25667	1618.221735	-40.231503	2561.52568	11.4099584	455.255608
PUC_p_128_128_1	256	65792	7643.74951	7780.95474	1.76334689	8031.96276	4.83335467	288.872159
PUC_p_128_128_2	256	65792	8008.76172	8499.00471	5.76823997	8631.699	7.21685588	285.379508
PUC_p_128_128_3	256	65792	7964.3916	7824.092644	-1.7931659	8053.78229	1.10992187	260.455434
PUC_p_128_128_4	256	65792	7530.24023	8046.52575	6.41625383	8261.84318	8.8552025	264.257492
PUC_p_128_128_5	256	65792	7232.15527	8038.88871	10.0353851	8225.87386	12.0804014	300.269715

Branch and Cut Results

Instance	|V|	|E|	Reduced(%)	Value	Nodes	Depth	T(s)
PUC_p_20_20_1	40	1640	0	347.490987*	1	0	0.056755
PUC_p_20_20_2	40	1640	12.31707	381.249147*	183	28	1.298876
PUC_p_20_20_3	40	1640	4.451218	391.298766*	149	17	1.265832
PUC_p_20_20_4	40	1640	31.585365	399.452595*	19719	206	241.501502
PUC_p_20_20_5	40	1640	52.256096	366.812088*	2651	276	45.415632
PUC_p_22_22_1	44	1980	20.050507	459.009082*	43247	199	1007.6632
PUC_p_22_22_2	44	1980	12.222221	413.546758*	101	22	0.938931
PUC_p_22_22_3	44	1980	28.585861	433.645458*	49785	408	662.044358
PUC_p_22_22_4	44	1980	12.828285	461.944699*	1033	44	10.566099
PUC_p_22_22_5	44	1980	3.484848	448.584133*	23	11	0.177069
PUC_p_24_24_1	48	2352	31.590134	460.297172*	287	55	4.557817
PUC_p_24_24_2	48	2352	6.802719	477.413172*	87	43	0.787301
PUC_p_24_24_3	48	2352	13.095238	418.466013*	1205	119	10.378795
PUC_p_24_24_4	48	2352	0	459.073238*	1	0	0.039044
PUC_p_24_24_5	48	2352	4.081635	480.622508*	137	30	0.58351
PUC_p_26_26_1	52	2756	8.164009	573.995609*	56879	112	414.938741
PUC_p_26_26_2	52	2756	0	515.261508*	1	0	0.042523
PUC_p_26_26_3	52	2756	17.162552	594.352813*	579	70	6.805739
PUC_p_26_26_4	52	2756	31.494919	472.589049*	36317	412	779.762809
PUC_p_26_26_5	52	2756	11.393326	585.795691*	2475	146	39.569431

Branch and Cut Results

Instance	|V|	|E|	Reduced(%)	Value	Nodes	Depth	T(s)
PUC_p_28_28_1	56	3192	19.956139	640.926754*	199173	304	3602.43746
PUC_p_28_28_2	56	3192	67.543861	631.736623*	59347	1263	3158.64448
PUC_p_28_28_3	56	3192	13.972427	594.231017*	6937	160	111.45033
PUC_p_28_28_4	56	3192	22.744362	570.737114*	5959	185	97.521915
PUC_p_28_28_5	56	3192	0	544.909304*	1	0	0.044575
PUC_p_30_30_1	60	3660	8.142075	639.400587*	4451	159	78.544597
PUC_p_30_30_2	60	3660	25	722.46814*	32121	303	1278.15948
PUC_p_30_30_3	60	3660	16.065575	732.901971*	1655	116	37.608023
PUC_p_30_30_4	60	3660	54.726776	668.176955*	12699	622	435.241974
PUC_p_30_30_5	60	3660	33.142075	706.649792*	98733	605	2353.94642
PUC_p_32_32_1	64	4160	41.89904	788.831899*	185595	988	11290.9763
PUC_p_32_32_2	64	4160	8.004807	748.655533*	1201	81	26.319408
PUC_p_32_32_3	64	4160	9.783653	754.848532*	41	16	1.233927
PUC_p_32_32_4	64	4160	43.4375	835.28175	160472	709	12000
PUC_p_32_32_5	64	4160	23.79808	789.848633*	213339	402	5897.59998

Branch and Cut Results

Instance	|V|	|E|	Reduced(%)	Value	Nodes	Depth	T(s)
PUC_p_40_40_1	80	6480	39.521606	1140.72296	184791	1095	12000
PUC_p_40_40_2	80	6480	58.410492	1211.5083	99994	1233	12000
PUC_p_40_40_3	80	6480	78.533951	1243.62585	118148	1539	12000
PUC_p_40_40_4	80	6480	68.456787	1158.59005	150530	1527	12000
PUC_p_40_40_5	80	6480	15	1122.354712	1399	167	93.95155
PUC_p_48_48_1	96	9312	9.80455	1373.49672	336208	308	12000
PUC_p_48_48_2	96	9312	90.850517	1614.87205	16407	1207	12000
PUC_p_48_48_3	96	9312	75.042953	1727.82459	21347	1613	12000
PUC_p_48_48_4	96	9312	75.633591	1701.77234	15192	1789	12000
PUC_p_48_48_5	96	9312	8.601807	1422.951512	6803	240	602.135652
PUC_p_64_64_1	128	16512	82.412788	2724.85596	6012	2549	12000
PUC_p_64_64_2	128	16512	9.895836	2195.65564	34138	891	12000
PUC_p_64_64_3	128	16512	40.915699	2536.14063	17577	949	12000
PUC_p_64_64_4	128	16512	82.909401	2546.45801	6244	2570	12000
PUC_p_64_64_5	128	16512	7.249275	2269.256668	3193	321	687
PUC_p_128_128_1	256	65792	99.516655	7643.74951	282	281	12000
PUC_p_128_128_2	256	65792	99.293228	8008.76172	298	297	12000
PUC_p_128_128_3	256	65792	100	7964.3916	364	363	12000
PUC_p_128_128_4	256	65792	93.753036	7530.24023	482	481	12000
PUC_p_128_128_5	256	65792	66.51416	7232.15527	843	841	12000

Conclusions

- 2D-unwrapping, a challenging problem, was addressed
- a New Model was proposed
  - Experiments showed that the new model MSFBC better captures the essence of unwrapping, at least in 2D.
  - For quite a few examples, less valued solutions implied in better unwrappings
- A Generalization of the Classical Steiner Problem in Graphs was described.
- The MSFBC appears as an interesting problem to study, already with a critical application in many areas, including oil and gas.

Conclusions

- 2D-unwrapping, a challenging problem, was addressed
- a New Model was proposed
  - Experiments showed that the new model MSFBC better captures the essence of unwrapping, at least in 2D.
  - For quite a few examples, less valued solutions implied in better unwrappings
- A Generalization of the Classical Steiner Problem in Graphs was described.
- The MSFBC appears as an interesting problem to study, already with a critical application in many areas, including oil and gas.

Conclusions

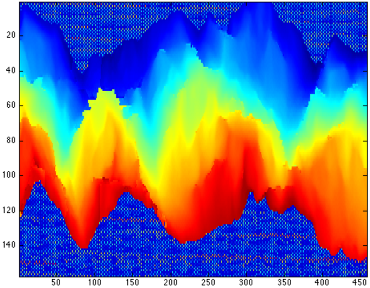
- 2D-unwrapping, a challenging problem, was addressed
- a New Model was proposed
    - Experiments showed that the new model MSFBC better captures the essence of unwrapping, at least in 2D.
    - For quite a few examples, less valued solutions implied in better unwrappings
- A Generalization of the Classical Steiner Problem in Graphs was described.
- The MSFBC appears as an interesting problem to study, already with a critical application in many areas, including oil and gas.

Future Work

- Resolution Methodology
  - More carefully tailored instances
  - More tests of the proposed models and algorithms
  - Test a Column Generation formulation: contrary to the SPG the MSFBC has a natural decomposition
- Applications
  - The MSFBC naturally models 3D-unwrapping: testing and exploring its capabilities may lead to improvements in the state-of-the-art, specially for Oil and Gas

Future Work

- Resolution Methodology
  - More carefully tailored instances
  - More tests of the proposed models and algorithms
  - Test a Column Generation formulation: contrary to the SPG the MSFBC has a natural decomposition
- Applications
  - The MSFBC naturally models 3D-unwrapping: testing and exploring its capabilities may lead to improvements in the state-of-the-art, specially for Oil and Gas



Merci!