Spanning Trees for Colored Point Sets

Carleton: A. Biniaz, P. Bose, K. Crosbie, A. Maheshwari, P. Morin, M. Smid
U. Ottawa: J.L. De Carufel
UC Irvine: D. Eppstein

Outline

Preliminaries

Problem Definition

MinST of $K_{R,B}$

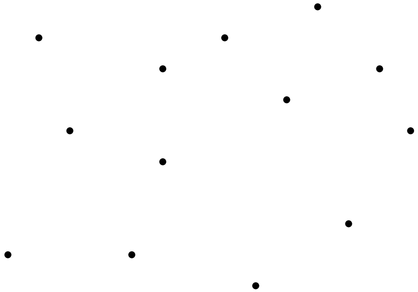
$O(n \log^3 n) \to O(n \log n)$

Plane Spanning Trees

Open Problems

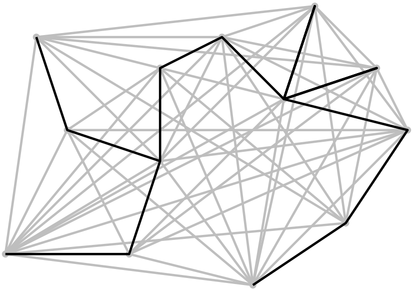
VD, DT, and MST

A Point Set P:



VD, DT, and MST

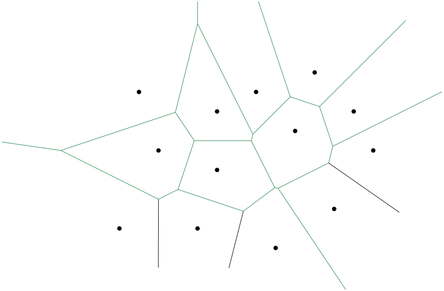
Complete Graph $K_{|P|}$:



Problem: Compute MST of $K_{|P|}$

VD, DT, and MST

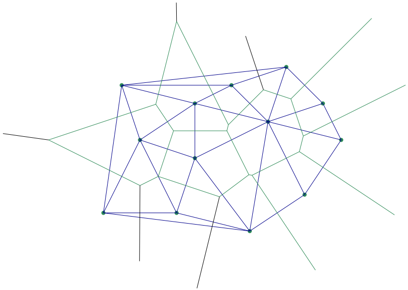
Voronoi Diagram of P:



$O(n\log n)$ [SH75]

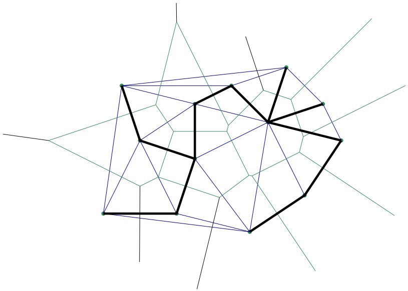
VD, DT, and MST

Delaunay Triangualtion of P:

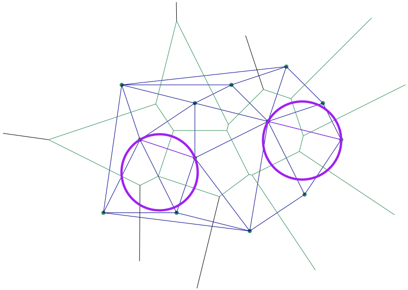


VD, DT, and MST

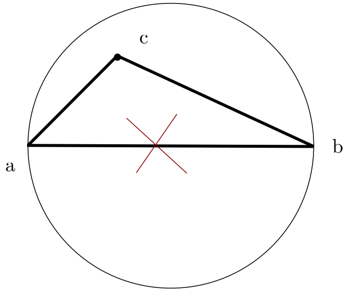
Euclidean MST of Complete Graph $K_{|P|}$:



Empty Circle property of Edges in $DT(P)$



$MST \subseteq DT$



All edges of MST satisfy the empty-circle property.
All edges satisfying the empty-circle property $\in DT$.
$\Rightarrow MST \subseteq DT$.

Algorithm for computing MST

Algorithm:

1. Compute Delaunay Triangulation of P.
   Let G be the graph corresponding to $DT(P)$.
2. Compute MST of G by executing any of
   Kruskal or Prim or Borůvka's algorithm.

Time Complexity:

- Let $n = |P|$.
- $DT(P)$ is plane and has $O(n)$ edges.
- MST can be computed in $O(n \log n)$ time.

Theorem: Euclidean MST of n points in plane can be computed in $O(n \log n)$ time.

Problem Definition

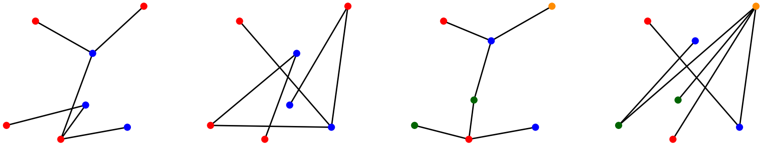
Input: P= A set of n coloured points in plane.

Graph: $G = (V, E)$

1. $V = P$.
2. Edge between every pair of points of different colors.
3. Weight of $e = (uv) \in E$ is $|uv|$.

Output: A minimum/maximum Euclidean Spanning Tree of G.

Minimum and Maximum Spanning Trees



Min-2-ST Max-2-ST Min-4-ST Max-4-ST

Note:
G is complete bipartite graph $K_{R,B}$ for Min/Max-2-ST problems.
G is complete multipartite graph for Min/Max-4-ST problems.

MST for Bichromatic Point Sets

Input: $P = R \cup B$ a collection of Red and Blue points.

Graph: A Complete Bipartite graph $G = K_{R,B}$.

Output: Euclidean Minimum Spanning Tree of G.

Algorithm: Execute Borůvka's MST algorithm on G.

Borůvka's algorithm

Input: G

Output: $T := MST(G)$

1. $T := \emptyset$
2. For each vertex v, find the edge with minimum weight incident on v (say vw).
3. $T := T \cup \{vw\}$.
4. Identify v and w.
5. Repeat Steps 2-4 till G has more than one vertex.

Illustration of Borůvka's algorithm

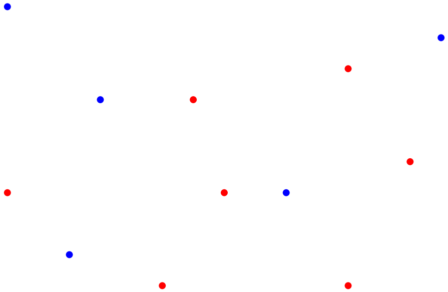


Illustration of Borůvka's algorithm

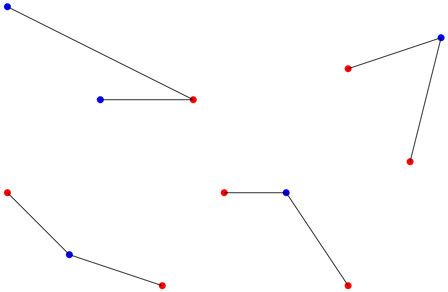


Illustration of Borůvka's algorithm

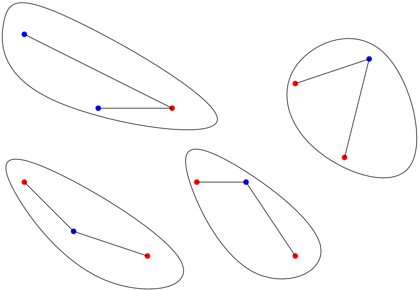


Illustration of Borůvka's algorithm

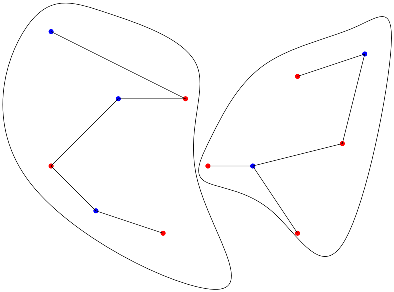
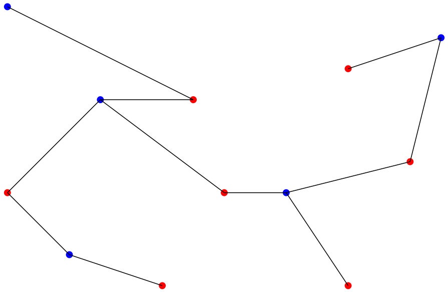


Illustration of Borůvka's algorithm



Borůvka's algorithm

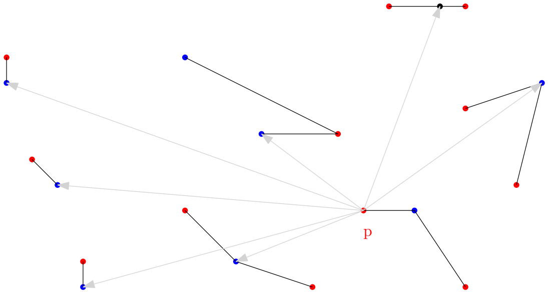
Input: G

Output: $T := MST(G)$

1. $T := \emptyset$
2. For each vertex v, find the edge with minimum weight incident on v (say vw).
3. $T := T \cup \{vw\}$.
4. Identify v and w.
5. Repeat Steps 2-4 till G has more than one vertex.

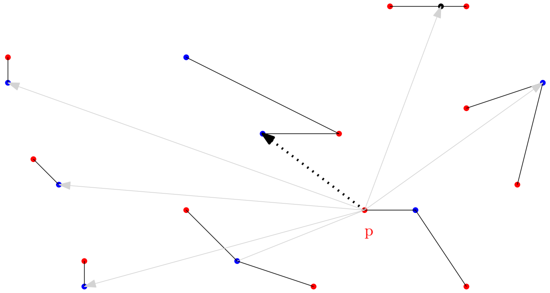
Observation: Steps 2-4 execute at most $O(\log n)$ times. Why?
Question: How to use geometry to execute Step 2 efficiently?

Nearest unrelated points problem



$NN(p)$ = Nearest blue point to p exterior to its component

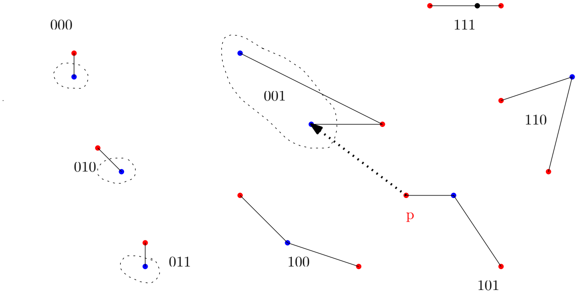
Nearest unrelated points problem



$NN(p) =$ Nearest blue point to p exterior to its component

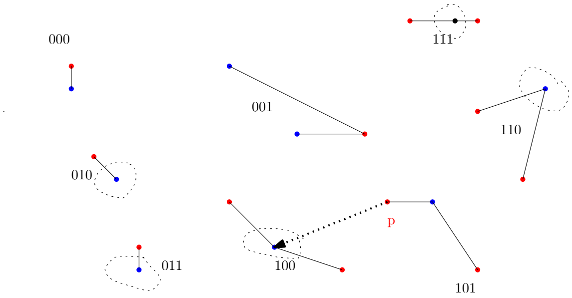
Nearest unrelated points problem

Find $NN(p)$ among blue points corresponding to components labelled with significant bit = 0



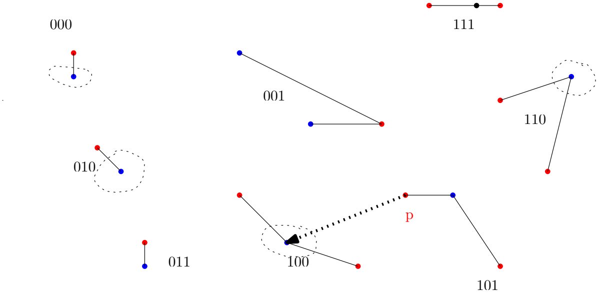
Nearest unrelated points problem

$NN(p)$ among blue points with components with middle bit $= 1$



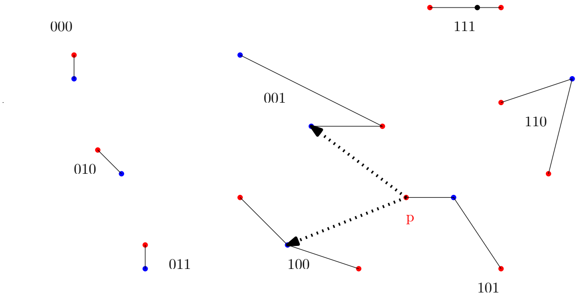
Nearest unrelated points problem

$NN(p)$ among blue points with components with LSB $= 0$

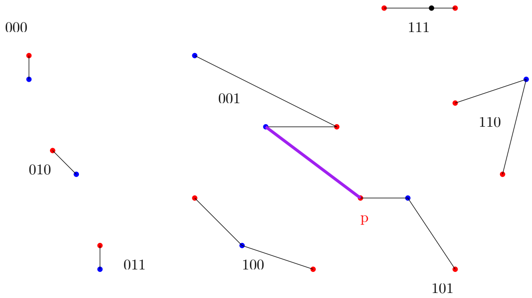


Nearest unrelated points problem

Choose the nearest among $O(\log n)$ potential nearest neighbours:



Nearest unrelated points problem



Complexity Analysis

For each phase (Steps 2-4) of Borůvka's algorithm, we compute

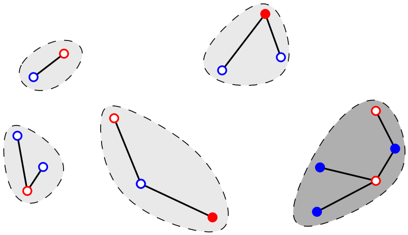
1. $O(\log n)$ Voronoi diagrams of subsets of blue points.
2. For each red point p, we perform $NN(p)$ queries in blue-Voronoi diagrams.
3. We do the same computation where the roles of red and blue colors are reversed.

Complexity of each phase $= O(n \log^2 n)$.

Total Complexity $= O(n \log^3 n)$.

Bi-chromatic Closest Pair Problem

For each component, find the closest red-blue pair, where one point is outside the component.
Problem: Find $BCP(B_i, R \setminus R_i)$ and $BCP(R_i, B \setminus B_i)$.



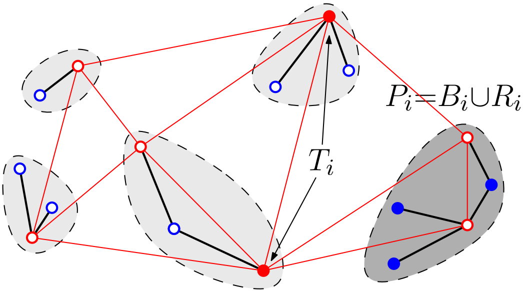
Algorithm for computing $BCP(B_i, R \setminus R_i)$

Input: A set of $R \cup B$ points, partitioned among components $P_i = R_i \cup B_i$.
Output: For each $P_i = R_i \cup B_i$, $BCP(B_i, R \setminus R_i)$ = its nearest red point in $R \setminus R_i$

1. Construct $DT(R)$
2. For each component i do
   2.1 Compute $T_i = \{p \in R \setminus R_i :$ in $DT(R)$, p is adjacent to a point in $R_i\}$
   2.2 Construct $DT(B_i \cup T_i)$
   2.3 $BCP(B_i, R \setminus R_i)$ = the endpoints of a shortest red-blue edge in $DT(B_i \cup T_i)$

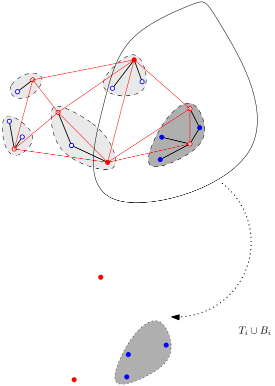
Computation of $BCP(B_i, R \setminus R_i)$

T_i from $DT(R)$:



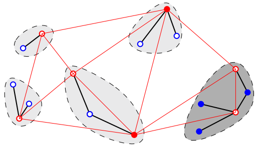
Computation of $BCP(B_i, R \setminus R_i)$

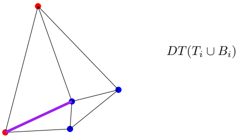
Isolating $T_i \cup B_i$:



Computation of $BCP(B_i, R \setminus R_i)$

$BCP(B_i, R \setminus R_i)$ from $DT(T_i \cup B_i)$:



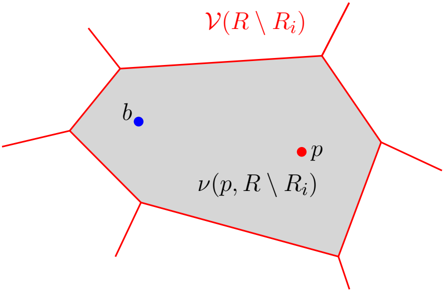


Proof of Correctness

Why is $BCP(B_i, R \setminus R_i) = BCP(B_i, T_i)$?

Let $b \in B_i$ and $p \in R \setminus R_i$, such that $\{b, p\} = BCP(B_i, R \setminus R_i)$.

Why $DT(R)$ has an edge from some point $q \in R_i$ to p?

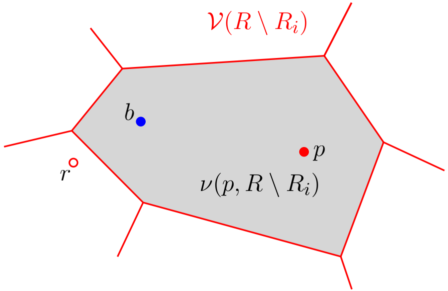


Proof of Correctness

Why is $BCP(B_i, R \setminus R_i) = BCP(B_i, T_i)$?

Let $b \in B_i$ and $p \in R \setminus R_i$, such that $\{b, p\} = BCP(B_i, R \setminus R_i)$.

Why $DT(R)$ has an edge from some point $q \in R_i$ to p?

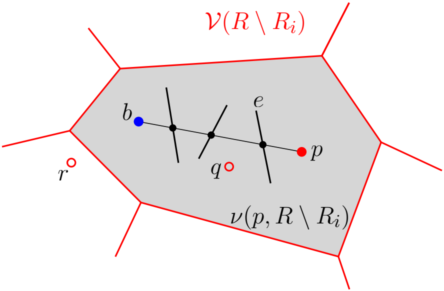


Proof of Correctness

Why is $BCP(B_i, R \setminus R_i) = BCP(B_i, T_i)$?

Let $b \in B_i$ and $p \in R \setminus R_i$, such that $\{b, p\} = BCP(B_i, R \setminus R_i)$.

Why $DT(R)$ has an edge from some point $q \in R_i$ to p?



Complexity Analysis

For $BCP(B_i, R \setminus R_i)$ computation:

- $\sum_i |T_i| = \sum_{r \in R} degree(r)$ in $DT(R) = O(|R|)$
- $\sum_i |B_i| = O(|B|)$
- $\forall i, DT(T_i \cup B_i)$ can be computed in $O(n \log n)$ time

$\Rightarrow \forall i, BCP(B_i, R \setminus R_i)$ can be computed in $O(n \log n)$ time
- Same holds for $BCP(R_i, B \setminus B_i)$.

Borůvka's algorithm has $O(\log n)$ phases.
$\Rightarrow$ MST of $R \cup B$ can be computed in $O(n \log^2 n)$ time.

MinST of $B \cup R$

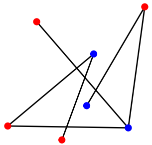
- Computation of BCP uses $DT(T_i \cup B_i)$.
- $DT(T_i \cup B_i)$, for each i, can be computed by the algorithms for maintaining DT under merge/split operations.
- Apply Dynamic Data Structures + Amortization Arguments:

Theorem: MST of $R \cup B$ can be computed in $\Theta(n \log n)$ time.

MaxST of $B \cup R$

I: For every edge (r, b) in MaxST, either $r \in CH(R)$ or $b \in CH(B)$.

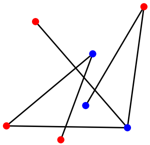
II: Longest edge between R and B has an endpoint in $CH(R)$ and an endpoint in $CH(B)$.



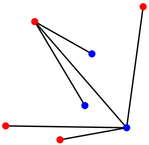
Borůvka's algorithm + above observations + fartherst-point VD + $BFP(B_i, R \setminus R_i)$ and $BFP(R_i, B \setminus B_i)$ leads to
Theorem: MaxST of $R \cup B$ can be computed in $\Theta(n \log n)$ time.

Plane Max ST

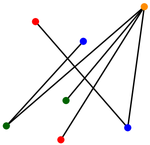
Input: A set of coloured points P (and a complete multi-partite graph G on P).
Output: A Plane Maximum Spanning Tree of G.



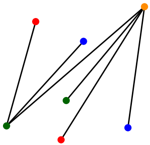
MaxBST



MaxBPST



Max-4-ST

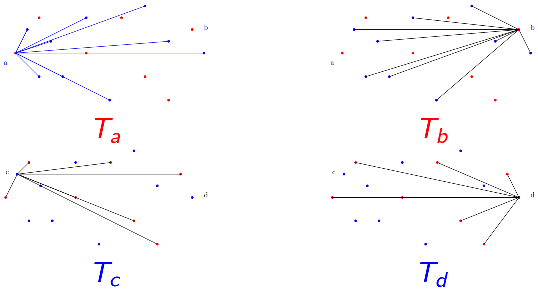


Max-4-PST

Results on Plane Min/Max ST

#colors	Min/Max	Exact	Approx.	Reference
	Min	$\Theta(n \log n)$		SH75
	Max	NP-Hard?	2	ARS93
			1.993	DT10
			1.989	BBdCCEMS17
2	Min	NP-Hard	$\sqrt{n}$	BvKLLMSV09
	Max		4	BBdCCEMS17
3	Max		6	BBdCCEMS17
≥ 4	Max		8	BBdCCEMS17

4-approx for Bichormatic Plane Max ST



Output: Max of $\{T_a, T_b, T_c, T_d\}$

Analysis

- $T^* =$ an optimal tree.
- Root T^* at a red vertex r.
- Orient edges away from r.
- $E_r =$ Edges from red vertices to blue vertices.
- $E_b =$ Edges from blue vertices to red vertices.
- $Cost(T^*) = Cost(E_r) + Cost(E_b)$
- Let $Cost(E_r) \leq Cost(E_b) \Rightarrow Cost(T^*) \leq 2Cost(E_b)$
- Let ab be a diameter of red points.
- We show: For each blue vertex b', its distance to any red vertex r' satisfies $|b'r'| \leq |b'a| + |b'b|$.

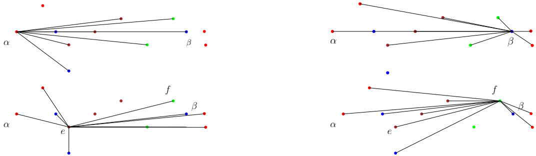
$\Rightarrow Cost(E_b) \leq Cost(T_a) + Cost(T_b)$

$\Rightarrow Cost(T^*) \leq 4Max(T_a, T_b, T_c, T_d)$

8-approx for multi-coloured Plane Max ST

We compute 8 different stars.

- $\alpha\beta$ = chromatic diameter of point set P
- ab = red diameter; cd = blue diameter
- ef = chromatic diameter of $(P \setminus \{Red \cup Blue\})$.



Output: Max of $\{T_\alpha, T_\beta, T_a, T_b, T_c, T_d, T_e, T_f\}$

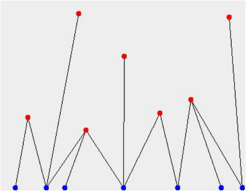
Plane Bi-chromatic ST for Convex Point Sets

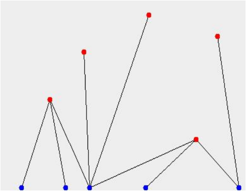
- Assume $P = R \cup B$ and $|R| \geq |B|$

- Min/Max Spanning Plane Tree in $O(|R|^2|B|)$ time.

- Based on Dynamic Programming.

Plane Bi-chromatic ST for Semi-collinear Point Sets





$O(|R|^3|B|^2)$ dynamic programming algorithm for Min/Max/Bottleneck/... Plane bi-chromatic Spanning Trees.

Problems to Ponder

- For k-coloured versions, Min-k-ST is computed in $O(n \log n \log k)$ time. Optimal for $k = 2$ and $k = n$. What about other values of k?

- For plane Min/Max ST

1. $O(\sqrt{n})$-approximation for Min ST is known in $K_{|P|}$. Improvements?
2. Is ratio of $\frac{wt(\text{Plane Min ST})}{wt(\text{MST})} \leq 1.5$ in $K_{R,B}$?
3. Characterization of edges in $K_{R,B}$ in Plane Min/Max ST?
4. Given a Plane ST of $K_{R,B}$, is it optimal?
5. Semi-collinear case: Red points on both sides of the line?

Thanks a lot for listening
Questions/Comments?

February(avg): Ottawa: -9.2C; Snowfall 35cm