Efficient strategy for Maximal Points Verification under Uncertainty

George Charalambous and Michael Hoffmann

Department of Computer Science, University of Leicester
gc100@mcs.le.ac.uk, mh55@mcs.le.ac.uk

Abstract. The study of algorithms that handle imprecise input data for which precise data can be requested is an interesting area. In the *verification under uncertainty* setting, an algorithm is given a set of imprecise input data and a set with the corresponding assumed precise data. An *update request* on some input reveals its precise value and it is *verified* if this matches the corresponding assumed value. The aim of an algorithm under this setting is to perform the smallest number of such verified requests such that it allows for a solution of the underlying problem to be calculated. In the classical problem of computing maximal points, a set of points in 2-dimensional space is given and an algorithm must find all points that are not dominated by other points. Considering the verification under uncertainty setting for the maximal points problem (MPV) an algorithm is given for each input point a range of possible locations, together with the corresponding assumed precise location. We call these ranges the *areas of uncertainty*. An update request can be made on an area of uncertainty which reduces the possible locations to a single element, and if this is the same as the assumed precise location then the update is verified. The objective here is to find a minimal set of update requests such that, given the updates verify the assumed data, the set of all maximal points can be computed based on the information of the areas of uncertainty and the results of the updates. It was shown that the unrestricted problem is NP-hard. In this paper we propose a new restriction on the accepted input data and introduce an algorithm which runs in polynomial time with this restriction in place. The restriction is that each area of uncertainty A_i is composed of all points produced by the direct product of possible x and y coordinate values.

1 Introduction

Precise input data may not always be easily available. Precision may be expensive, time consuming, require large computational power or for other reasons even infeasible. Furthermore there are cases where data might change over time, but only slightly such that the new data is guaranteed to be somewhat close to the old data. In many cases obtaining an estimate of an input data item, i.e. a range in which it is guaranteed to lie, may be more cost-efficient and enough to work with. If a collection of estimates for items is insufficient, then obtaining the

precise value in some of these items might be enough for producing an outcome. Our research focuses around this setting.

Nowadays more and more information is available. With a flood of sensors connected to a network, such as GPS-enabled mobile phones, up-to-date readings of these sensors are generally available. Algorithms that perform based on such information might not have the precise data available to them, as for example in some situations the traffic of collecting all such information would cause a problem on its own. In other situations, obtaining the up-date information of all sensors is costly in time and battery power or even other charges may occur. A practical solution is to work with slightly out of date data where possible and only request up to date information where needed. For example a sensor may automatically send its current measurement if this exceeds some predefined bounds of the latest send one. Hence based on the last send information a possible band or area is known where the current measurement of the sensor is within.

Dealing with scenarios of uncertainty where queries/updates are available to get more precise information is a relatively new topic. Most classical algorithms produce an output with an assumed precise input. For a problem having this setting of uncertainty, each input might be given in a rough (uncertain) form and the precise value of the input can be obtained by requesting to update it at a cost. An algorithm is asked to act based on the current, possibly uncertain, information and if the current information is insufficient then *updates* must be considered. The task of the algorithm is to request updates until the information allows for a solution of the underlying problem to be calculated. We call the set of these updates an *update solution* and if it is also of minimal size then we refer to it as an *optimal update solution*. As each update is entitled to a predefined uniform cost, and generally the less total cost the better, minimizing the use of updates is desired.

In the *adaptive online* under uncertainty setting an algorithm initially knows only the uncertain information and performs updates one by one (determining the next update based on the information from previous updates) until it has obtained sufficient information to determine a solution. Algorithms are typically evaluated by competitive analysis, comparing the number of updates they make with the minimum number of updates that, in hindsight, would have been sufficient to determine a solution.

In the *verification* under uncertainty setting the algorithm is not only given the uncertain information, but also an assumed set of precise input data. If after an update operation the precise value obtained is equal to the corresponding assumed value, we say that the update *verifies* the assumed value. The objective of an algorithm in the verification setting is to produce an optimal update solution, under the assumption that the updates are verified.

In this paper we consider the Maximal Point Verification (MPV) problem, which falls in the verification setting of computing under uncertainty. The maximal point problem is a classical problem and many aspects of the problem have sparked interesting research. In its simple form without uncertainty it can be stated as follows: Given a set of 2-dimensional points, return all points that are

not dominated. A point is dominated if there exists a point with one coordinate greater and the other not lower. Introducing uncertainty to the problem the idea is that the coordinates of a point may not be known precisely but instead a region or *area of uncertainty* is given such that the point is guaranteed to lie within. Here the areas of uncertainty are restricted to be the direct product of some x and y coordinate values. In the verification problem we are further given the assumed precise location for each point, and the objective is to identify a minimal number of points such that the precise locations of these points together with the areas of uncertainty can produce the set of all maximal points. A formal definition is given for the MPV problem in Definition 2, and for the proposed restriction in Definition 4.

Our contribution is the new restriction to the MPV problem, as defined in Definition 4, and a polynomial time algorithm that solves it as shown in theorem 1. We further show that the 3 update competitive result for the online adaptive setting of the problem, as obtained by Bruce et al. in [2], carries over with this newly introduced restriction.

The effect of our result is significant for experimental evaluation of algorithms in the online adaptive setting of the maximal point problem under uncertainty. A polynomial time algorithm can be implemented and the results obtained can be used as a baseline to compare against online algorithms.

Related work:

Kahan [8] presented a model for handling imprecise but updateable input data. He demonstrated his model on a set of real numbers where instead of the precise value of each number an interval was given. That interval when updated reveals that number. The aim is to determine the maximum, the median, or the minimal gap between any two numbers in the set, using as few updates as possible. His work included a competitive analysis for this type of online algorithm, where the number of updates is measured against the optimal number of updates. For the problems considered, he presented online algorithms with optimal competitive ratio. Feder et al. [6] studied the problem of computing the value of the median of an uncertain set of numbers up to a certain tolerance. Applications of uncertainty settings can be found in many different areas including structured data such as graphs, databases, and geometry. The work presented in this paper mainly concerns the latter two areas.

Bruce et al. [2] studied the geometric uncertainty problem in the plane. Here, the input consists of points in the plane and the uncertainty information is for each point of the input an area that contains that point. They gave a definition of the Maximal Point under Uncertainty as well as the Convex Hull under Uncertainty. They presented algorithms with optimal competitive ratio for both problems. The algorithms used are based on a more general technique called *witness set algorithm* that was introduced in their paper.

Charalambous and Hoffmann [3] further studied the Maximal Point under Uncertainty problem for the verification setting, showing that the unrestricted problem is NP-Hard by a reduction from the Minimum Set Cover problem. Their proof uses disconnected areas of uncertainty containing at most two points. The

open question was raised if a restriction that captures real life examples could be put il place such that the problem becomes polynomial. This motivated the research to obtain the results for this paper.

In [5], Erlebach et al. studied the adaptive online setting for minimum spanning tree (MST) under two types of uncertainty: the edge uncertainty setting and the vertex uncertainty setting. In the latter setting, all vertices are points in the plane and the graph is a complete graph with the weight of an edge being the distance between the vertices it connects. The uncertainty is given by areas for the location of each vertex. For both settings, Erlebach et al. presented algorithms with optimal competitive ratio for the MST under uncertainty. The competitive ratios are 2 for edge uncertainty and 4 for vertex uncertainty, and the uncertainty areas must satisfy certain restrictions (which are satisfied by, e.g., open and trivial areas in the edge uncertainty case). A variant of computing under uncertainty where updates yield more refined estimates instead of exact values was studied by Gupta et al. [7].

In [4] Erlebach et al. further worked on MST for both edge and vertex uncertainty this time studying the verification problem. They give a polynomial-time optimal algorithm for the MST verification under edge uncertainty problem by relating the choices of updates to vertex covers in a bipartite auxiliary graph. For MST verification under vertex uncertainty they show that the problem is NP-hard even if the uncertainty areas are trivial or open disks. The proof is by reduction from the vertex cover problem for planar graphs with maximum degree 3.

Structure of the paper. In Section 2.1 we give formal definitions and preliminaries. In Section 3 we give in detail the algorithm and complete the proof of Theorem 1.

2 Preliminaries

2.1 Maximal Points under Uncertainty

Within the wider field of problems under uncertainty we consider the Maximal Point Verification (MPV) under uncertainty problem. We begin with establishing the classic problem.

The classic problem of finding maximal points in the plane has as input a set of points. All points discussed in the paper are points in the 2D plane, so a point p may be written in coordinate form (p_x, p_y). We say a point $p = (p_x, p_y)$ is *higher* than a point $q = (q_x, q_y)$ if $p_x \geq q_x$ and $p_y \geq q_y$ and $p \neq q$. This may also be written as $p > q$. We are interested in finding all points such that there does not exist a point higher, that is find all points in the set that are maximal. Note that this induces a partial order and leads to the following definition of a maximal point among a set of points.

Definition 1. *Let P be a set of points and p be a point in P. The point p is said to be maximal among P if there does not exist a point in P that is higher than p. Otherwise p is non-maximal among P.*

We use the uncertainty setting in the context of the Maximal Point problem. The idea is that the coordinates of a point p_i may not be known precisely but instead a region or *area of uncertainty* A_i is given such that $p_i \in A_i$. So we have the set of points $P = \{p_1, \ldots, p_n\}$ and the set of areas of uncertainty $\mathcal{A} = \{A_1, \ldots, A_n\}$ for each $1 \leq i \leq n$, with $p_i \in A_i$. If an uncertainty area A_i consists of a single possible location, we call this area *trivial* and $A_i = \{p_i\}$. The aim is to be able to produce the set of all points that are maximal among P based on the information of $\mathcal{A}$. If this is insufficient then *update operations* can be made on items of $\mathcal{A}$ which will reduce the uncertain areas to trivial size, i.e. $A_i = \{p_i\}$ for the selected update of i.

In the online adaptive setting the algorithm does not know the actual location of a point in P, until it chooses to update it. Therefore it takes as input the set $\mathcal{A}$, and this set is modified after every update by reducing the selected area to trivial size. A set of updates that reveal enough information to calculate the set of all maximal points is called, as stated earlier, an *update solution*. Updating all non-trivial areas would reveal the precise location for all points and therefore this would obviously be an update solution. For a given instance of a problem, we denote an update solution of minimal size also as an optimal update solution.

In the verification setting the algorithm is also given the set of assumed precise locations $P' = \{p'_1, \ldots, p'_n\}$ and therefore knows what the location of a point in P will be without modifying $\mathcal{A}$, under the assumption that $p'_i = p_i$ for an update $1 \leq i \leq n$. As such the aim here is to select a minimal number of updates such that when $\mathcal{A}$ is modified with the updates, the set of all maximal points can be calculated based on the information of the modified set $\mathcal{A}$.

Definition 2. *A Maximal Point Verification problem, MPV for short, is a pair* $(\mathcal{A}, P)$*, where* P *is a set of points and* $\mathcal{A}$ *is a set of areas for* P*. The aim is to identify a minimal set of areas from* $\mathcal{A}$*, that when updated verifies the maximal points among* P *as maximal based on the information of* $\mathcal{A}$ *and the results of the updates.*

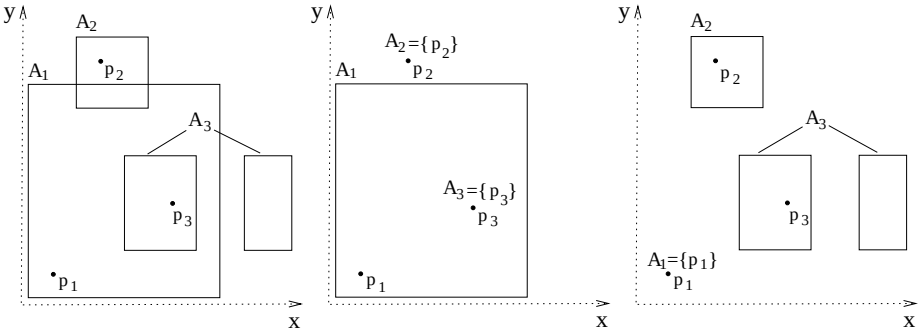

Fig. 1. Example of an MPV problem

Fig. 2. Updating A2,A3

Fig. 3. Updating A1

In the example shown in figure 1, the problem consists of three points (p_1, p_2 and p_3) and three areas A_1, A_2 and A_3 such that $\mathcal{A} = \{A_1, A_2, A_3\}$ and $P = \{p_1, p_2, p_3\}$. Note that the areas could be overlapping with each other and disconnected (as area A_3 is). We can see that points p_2 and p_3 are maximal among P since there is no other point in P that is higher. Also p_1 is non-maximal among P since at least p_2 or p_3 is higher. However without verifying any of the points in P (i.e. updating an area that the point is contained), each of the points in P may or may not be maximal based only on the areas of uncertainty. We can also observe that by updating every area except of A_1 the problem will not be solved since point p_1 could still potentially be maximal or non-maximal based on the initial areas of uncertainty and the information retrieved by the updates. (figure 2). Updating only A_1 on the other hand validates that p_1 is indeed non-maximal since A_1 is now trivial and wherever p_2 lies in A_2 (or p_3 lies in A_3), the point p_2 will always be higher. This brings us to figure 3 having performed one update so far that the problem could not have been solved without. Now we can also see that p_2 is maximal regardless where it lies in A_2 since there is no longer an area containing a potential location point higher than a potential location of p_2. It is therefore only left to show that p_3 is maximal and we now have a choice. Further updating either A_2 or A_3 will achieve this with a similar reasoning as before. So the problem has so far two potential update solutions $\{A_1, A_2\}$ and $\{A_1, A_3\}$ which are also optimal. We finish this example by noting that the set $\{A_1, A_2, A_3\}$ is also an update solution for the problem but it is not optimal since there exists another update solution of smaller size.

We will use the following notations:

An area A is said to be *maximal* among a set of areas $\mathcal{A}$ if there does not exist a point in any areas in $\mathcal{A} - A$ that is higher than any point in A. Similarly, an area A is said to be *non-maximal* among a set of areas $\mathcal{A}$ if for every point $p \in A$ there is an area in $B \in \mathcal{A}$ such that every point in B is higher than p.

We also note that an area might be neither maximal nor non-maximal among a set of areas, whereas a point is either maximal or non-maximal among a set of points as defined earlier. If this is the case then the set of maximal points cannot be calculated. In other words a problem is *solved* if and only if all areas in $\mathcal{A}$ are either maximal or non maximal among $\mathcal{A}$.

The areas which are neither maximal nor non-maximal can be further divided into the two following categories. Area A is *partly* among the set $\mathcal{A}$ if A is neither maximal nor non-maximal and further contains a point p such that there does not exist a point in any areas in $\mathcal{A} - A$ that is higher than p. If such point p does not exist then area A is *dependent* among the set $\mathcal{A}$.

For further convenience we say an area A *interferes* with an area B if there exists a point in A higher than a point in B.

2.2 New Model

The problem of finding maximal points under uncertainty was introduced by Bruce et al. in [2] and an update optimal strategy was given for the restricted online problem, as also it was shown that a constant update competitive ratio is

impossible for unrestricted areas. The proposed restriction they gave is to have only areas which are either trivial or closures of connected, open areas.

Although the restricted problem as studied by Bruce et al. yields nice results, it does not capture scenarios where the uncertain information about an item may contain disconnected values. For example when comparing products a product may have an uncertain price in disconnected ranges depending on availability from different sellers, time of purchase, combination of products purchased, etc. As the restriction by Bruce et al. does not cover situations like this, we wish to have a model where the categorized information does not have to be continuous but instead it can be gradually moving.

Further study of the problem by Charalambous and Hoffmann [3] has shown that the verification problem is NP-hard if there are no restrictions on the areas of uncertainty. This is shown to be the case even if the areas of uncertainty contain at most two points.

The results of the work of Bruce et al. and Charalambous and Hoffmann gave rise to even further study of the verification problem. There was the expectation of restricting the accepted input set $\mathcal{A}$ such that: an algorithm can solve the verification problem in polynomial time, an algorithm can solve the online problem with the 3-update competitive strategy of Bruce et al., and also accept other types of uncertain areas which the restriction by Bruce et al. was unable to do so.

Although uncertain information about an item may contain disconnected values, usually this information is categorized such that the properties of the item are independent of each other. For example when comparing products the price of a product should not affect its rating. With the maximal points under uncertainty in mind, independence between the x and y axes can be modelled as constructing an area of uncertainty from the direct product of two sets, one contain x coordinate values and another containing the y coordinate values. This further allows disconnected values. This kind of restriction captures problems such as finding the top products from a list where each product is given price and rating in independent ranges, but also each range may have disconnected values. It is an interesting idea for a new model for studying the maximal points under uncertainty problem, provided that efficient algorithms for both the online and verification settings are possible.

Following the above we introduce the following restriction for the accepted input areas for the verification problem. Given that we are still studying the problem in 2-dimensional euclidean space only, a point has two coordinates namely x and y. The areas of uncertainty may only be composed of all points made up by the direct product of possible x and y values. Formally:

Definition 3. *The input pair $(\mathcal{A}, P)$ of the MPV problem is such that $\mathcal{A} = \{A_1, \ldots, A_n\}$ and $A_i = \{A_i^x \times A_i^y\}$ where A_i^x is a set of x values, A_i^y is a set of y values and $1 \leq i \leq n$.*

Given for example uncertain values for x-coordinate $A_i^x = \{1, [2, 4]\}$ and y-coordinate $A_i^y = \{2, [4, 6]\}$ the area of uncertainty A_i can be depicted as shown in the following figure.

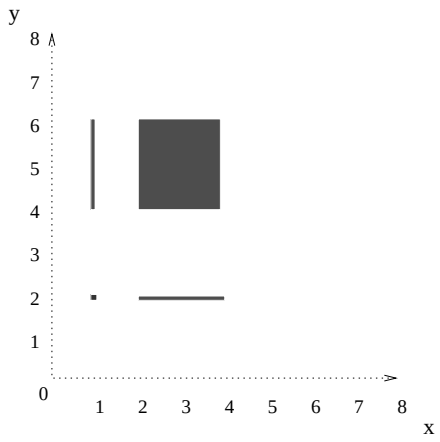

Fig. 4. Example of area $A_i = \{1, [2, 4]\} \times \{2, [4, 6]\}$

We now show that the 3-update competitive witness algorithm result of Bruce et al in [2] also applies for the online setting problem of our model. It is interesting that our restriction allows some disconnected areas, whereas this is not the case in the restriction proposed by Bruce et al. They have the restriction for areas of uncertainty such that they are either trivial or closures of connected open areas. Their Lemma 4 makes use of the restriction and shows that if a vertical or horizontal line splits two areas then they cannot both be maximal among $\mathcal{A}$. Furthermore in their Lemma 5 they establish the witness set of size 3. The rest of the reasoning used by Bruce et al is elementary to follow with our model in mind as no use of a restriction is necessary. Therefore we show a similar reasoning below applied to our restriction that is equivalent to their reasoning for those 2 Lemmas, and also a slightly different definition when a line *splits* an area. This suffices to establish that their result applies here as well. We have that:

Definition 4. *Let A be a non-trivial area. If l is a vertical line at X and there exist points $a, b \in A$ such that $a^x \geq X$ and $b^x < X$ then we say l splits A. Similarly if l is a horizontal line at Y and there exist points $a, b \in A$ such that $a^y \geq Y$ and $b^y < Y$ then we say l splits A.*

If there exists an area that is partly among $\mathcal{A}$ then as by Lemmas 2 and 3 of Bruce et al. there exists a witness set of size at most 2. We now prove the following two lemmas using similar reasoning as in Bruce et al. Lemmas 4 and 5, but using our version of the Lemma 1:

Lemma 1. *A horizontal or vertical line can split at most one area that is maximal among $\mathcal{A}$.*

Proof. Let l be a vertical line at X and let A_i, A_j be areas maximal among $\mathcal{A}$. Further let l split A_i, A_j such that there exist points $a_i, b_i \in A_i$ and $a_j, b_j \in A_j$ with $a_i^x, a_j^x \geq X$ and $b_i^x, b_j^x < X$.

We have that either $a_i > b_i$ or $a_i \not> b_i$. If $a_i > b_i$ then we choose point $c_i = a_i$. Otherwise we choose point $c_i = (a_i^x, b_i^y)$. As by the restriction of direct products we have that c_i must exist in A_i, and we note in both cases we have that $c_i > b_i$. Further as $c_i > b_i$ and $c_i^x > b_i^x$ then we have $c_i^y \geq b_i^y$.

With the same reasoning we choose $c_j \in A_j$ such that $c_j > b_j$, $c_j^x > b_j^x$ and $c_j^y \geq b_j^y$.

We now have that either $c_i^y \geq c_j^y \geq b_j^y$ or $c_j^y \geq c_i^y \geq b_i^y$. As c_i, c_j lie on or to the right of l and b_i, b_j lie to the left of l, then it follows that either $c_j > b_i$ or $c_i > b_j$. This is a contradiction as both A_i, A_j are maximal among $\mathcal{A}$ and therefore there must be no point in areas $\mathcal{A} - A_i$ dominating b_i, and no point in areas $\mathcal{A} - A_j$ dominating b_j. As such if l splits A_i and A_j, one of them cannot be maximal among $\mathcal{A}$.

In a similar way if l is a horizontal line at Y and splits areas A_i, A_j, then there exist points $c_i, b_i \in A_i$ and $c_j, b_j \in A_j$ such that $c_i^x \geq c_j^x \geq b_j^x$ or $c_j^x \geq c_i^x \geq b_i^x$ holds. As c_i, c_j would lie on or above of l and b_i, b_j below l, then it follows that either $c_j > b_i$ or $c_i > b_j$. Therefore A_i and A_j cannot be both maximal among $\mathcal{A}$.

□

Lemma 2. *If there are no party areas among $\mathcal{A}$ but there exists a dependent area, then there exists a witness set of size at most 3.*

Proof. Let A_i be a dependent area among $\mathcal{A}$ such that no other dependent area contains a point higher than A_i. Further let h_i be the highest point in A_i made by the upper limits of the coordinate sets that make the direct product of area A_i. We note that either $h_i \in A_i$ or $h_i \notin A_i$. Further let l_1 be the vertical line starting at h_i and going upwards, and l_2 be the horizontal line starting at h_i and going to the right. Let Q be the top right quadrant of l_1 and l_2 including l_1 and l_2. Further if $h_i \in A_i$ then let $h_i \notin Q$, and if $h_i \notin A_i$ then let $h_i \in Q$. In both cases any point in Q is higher than every point in A_i.

Since there is no other dependent area with a point higher than A_i as also there are no partly areas, every point that is in Q and contained in an area of $\mathcal{A}$ is maximal among P. Since A_i is dependent there exists an area A_j with a point in Q and a point not in Q. Since A_j contains a point in Q which can only be maximal among P, as also because there are no partly areas, A_j must be maximal among $\mathcal{A}$. Since A_j is maximal among $\mathcal{A}$, by Lemma 1, there are at most two areas that contain a point higher than A_i, one split by l_1 and one split by l_2. Let the other area be A_k. So we have to show whether A_i is maximal among $\mathcal{A}$ or not. If we need to show that A_i is maximal among $\mathcal{A}$ then we have to at least update both A_j and A_k, as otherwise there are points in A_j and A_k that remain in Q. If we need to show that A_i is not maximal among $\mathcal{A}$ then we have to at least update one of $\{A_j, A_k, A_i\}$ to show either $A_j > A_i$ or $A_k > A_i$. Therefore we have a witness set of size at most 3 with $\{A_j, A_k, A_i\}$.

□

This concludes the 3-update competitive witness algorithm proof for the online problem under the direct product restriction. The verification problem of

this new model brings us to the following theorem which we will prove at the end of the next section:

Theorem 1. *There exists a polynomial time algorithm for solving the Maximal Point Verification problem if each area of $\mathcal{A}$ is the direct product of x and y coordinate values.*

3 Verification Problem Algorithm

In this section we present a polynomial time algorithm that finds a solution for the MPV problem, under the restriction that each area of the input set $\mathcal{A}$ consist only of points produced by the direct product of a set of x and a set of y coordinate values.

As mentioned earlier the unrestricted problem is NP-Hard, even if the uncertain areas contain at most 2 points, and the proof for the NP-Hardness takes advantage of the dis-connectivity of the areas[*].

We will show that after the proposed algorithm has finished execution, all areas have a status of either maximal or non-maximal among $\mathcal{A}$ which is enough to distinguish all maximal points. Furthermore the set of updates performed will be of minimal size and hence solve the MPV problem. Finally we show that all steps of the algorithm can be done in polynomial time and hence theorem 1 is satisfied.

As introduced in the previous section, $P = \{p_1, \ldots, p_n\}$ is the set of points and $\mathcal{A} = \{A_1, \ldots, A_n\}$ is the set of areas with $p_i \in A_i$ for all $1 \leq i \leq n$. In contrast to the online setting, both sets P and $\mathcal{A}$ are given to the algorithm and therefore the algorithm can look up at the effect of including or not including a particular area to be part of an update solution. Recall that a set of areas that is a solution for the MPV problem may also be referred to as an optimal update solution, and an update solution, although it distinguishes all maximal points, is not necessarily a solution for MPV.

The algorithm runs in two phases over viewed as follows. In phase 1 the algorithm collects and performs updates as deemed necessary, hence changing the set $\mathcal{A}$, and also collects a series of choices between updates all of which every update solution would need to satisfy. The second phase then establishes an optimal way to satisfy all the choices collected from phase 1 and thus the set of updates from phase 1 together with the set of updates satisfying the choices together form an optimal solution for the problem.

As stated earlier areas are classified into maximal, non-maximal, partly and dependent among $\mathcal{A}$, as also their precise location is classified as either maximal or non maximal among P. Also during the run of the algorithm as updates are simulated, the classification of the areas and their precise location together with the information of $\mathcal{A}$ changes. To emphasize this we combine the two classifications into one, for example an area A_i is **D-M** if A_i is dependent among the current set $\mathcal{A}$ and the precise location $p_i \in A_i$ is maximal among P. In the same way we have the following types: **D-N** if A_i is dependent among $\mathcal{A}$ and

p_i is non-maximal among P, **P-M** if A_i is partly among $\mathcal{A}$ and p_i is maximal among P, and **P-N** if A_i is partly among $\mathcal{A}$ and p_i is non-maximal among P. Note that there are also areas of type **M-M** (A_i maximal among $\mathcal{A}$, p_i maximal among P) and **N-N** (A_i non-maximal among $\mathcal{A}$, p_i non-maximal among P) which the algorithm does not deal with. This is because, as mentioned earlier, when the current state of $\mathcal{A}$ does not contain partly or dependent areas no more updates are needed, and therefore we will show how each of the types D-M/D-N/P-M/P-N is dealt by the algorithm.

We also define the *self determined areas*. A self determined area A_i belongs to one of the four types D-M/D-N/P-M/P-N but it also has the following property which makes it mandatory to be included in every update solution. If after updating every area in $\mathcal{A}$ except from A_i itself, area A_i is partly among the new set $\mathcal{A}$ then A_i must be updated.

The outline of the algorithm is as follows:

```
Phase 1
    Update all self determined areas
        (eliminates all D-M,P-N areas and some D-N,P-M areas)
    Focus on remaining D-N areas: Discard areas, update clear
        better ones, record choices
    Focus on remaining P-M areas: Record choices
Phase 2
    Sort all maximal areas found within the choices by their
        precise location top-left to bottom-right
    Split all recorded choices in chains
    Update every second area in every chain
```

3.1 Phase 1

Overview.

The algorithm first establishes in 5 steps a set of updates that have to be in every update solution and flags areas that are not relevant for an update solution. Then a series of choices between updates are encountered in the following way. If out of the choices there is a clear better one, i.e. there exist an update which yields at least as much new information as the other matched choices, then this update is chosen and otherwise the choice between some updates is recorded. First phase simulates the selected updates and hence the areas in $\mathcal{A}$ change accordingly. All these updates are recorded and will be part of the output optimal update solution when the algorithm finishes execution.

Step 1.

For the first step we establish a set of updates that has to be a subset of any update solution as by the following Lemma:

Lemma 3. *Let $\mathcal{A} - A_i$ not be an update solution. Then every update solution must contain A_i.*

Let $A_i \in \mathcal{A}$ be an area such that after updating every area in $\mathcal{A}$ except A_i does not solve the problem. Formally, $\mathcal{A} - A_i$ is not an update solution and so any update solution must contain A_i as by Lemma 3. **The update of all such areas in this step is simulated by the algorithm**, and therefore set $\mathcal{A}$ is now modified with the updated data. As by Lemma 3, and as an effect of the action performed we have the following:

Remark 1. All updates performed in step 1 are needed in every update solution.

Lemma 4. *After the updates of step 1 there are no D-M and P-N areas in $\mathcal{A}$.*

Proof. Let's assume there exists an area A_i such that it is D-M after step 1. As A_i is dependent among $\mathcal{A}$ there exists an area A_j such that there is a point $q_j \in A_j$ and $q_j > A_i$. As the precise location point p_i is maximal among P, therefore $p_j \not> p_i$. Without updating A_j, though, area A_i cannot be identified as maximal among $\mathcal{A}$ since q_j remains in A_j. So $\mathcal{A} - A_j$ is not an update solution and therefore A_j must have been updated by step 1. This is a contradiction since A_j is non trivial as $q_j \in A_j$, $p_j \in A_j$ and $p_j \neq q_j$.

Let's assume that after step 1 there exists a P-N area A_i. As A_i is partly among $\mathcal{A}$ it cannot be trivial and it must contain a point q_i such that no point in any other area is higher than q_i. Furthermore the precise location point p_i is non-maximal among P. Without updating A_i itself, the area cannot be identified as non-maximal among $\mathcal{A}$ as the point q_i would remain in A_i. So $\mathcal{A} - A_i$ is not an update solution and therefore A_i must have been updated by step 1. This is a contradiction as A_i is non trivial.

□

Steps 2 and 3.

In the second and third steps we focus on dependent areas with non-maximal precise locations (D-N). Some D-N areas might have already changed their status after step 1. We now focus on the remaining D-N areas. Let A_i be a D-N area after step 1 is performed. As A_i is D-N then p_i is not maximal in P and therefore there exists p_j in P such that $p_j > p_i$. Note that for area A_i there may be another D-N area with precise location higher than A_i. We will only take action when this is not the case and we will show why this suffices for not leaving any D-N areas unhandled. At this stage there exist either one or two areas with precise location higher than every point in A_i. Formally:

Lemma 5. *Let A_i be a D-N area after the updates of step 1, such that there does not exist another D-N area with precise location higher than A_i. Then there exist either one or two areas with precise location higher than A_i.*

Proof. Recall the restriction proposed earlier for the uncertainty areas. As by the restriction area A_i is made by the direct product of a set A_i^x and a set A_i^y.

Let $h_i = (h^x, h^y)$ where h^x is the upper limit of A_i^x and h^y is the upper limit of A_i^y. We note the point h_i may or may not be in A_i and all other points in A_i are dominated by h_i. Further A_i does not contain a point that is maximal

among P as otherwise the instance cannot be solved without updating A_i and hence A_i would have been updated in step 1.

Case $h_i \in A_i$: As A_i does not contain a point that is maximal among P then h_i is also not maximal among P. As h_i is not maximal in P there exists a p_j in P that is higher than h_i and also $p_j > A_i$.

Case $h_i \notin A_i$: As h_i is made by the upper limits of A_i^x and A_i^y, either the upper limit of A_i^x is not in A_i^x or the upper limit of A_i^y is not in A_i^y, or both limits are not in their sets. As A_i does not contain any point that is maximal among P, there must exist a point p_j in P that is higher or equal than h_i. As $p_j \geq h_i$ and $h_i \notin A_i$ then also $p_j > A_i$.

By the condition of this lemma A_j, the area in which p_j is the precise location, cannot be a D-N area. Furthermore it cannot be a N-N area as p_j would be a non maximal point with respect to the current uncertain information and hence A_i would also be N-N instead of D-N since $p_j > A_i$. Also A_j cannot be P-N area by Lemma 4. So A_j is either an M-M, P-M or D-M area. In all cases the point p_j is maximal in P.

As area A_i is dependent, the area A_j must contain a point q'_j that is not higher than A_i. We now choose a point $q_j \in A_j$ with either $q_j = (q_j'^x, p_j^y)$ or $q_j = (p_j^x, q_j'^y)$ such that $q_j \not> A_i$ and $q_j < p_j$. As A_j is a product of possible x and y values point q_j exists in A_j. We note that as $q_j \not> A_i$, it is also not higher than h_i. The figure below shows these properties.

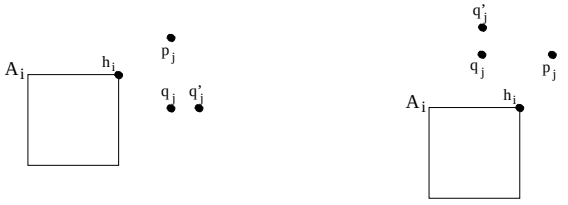

Fig. 5. D-N area A_i with $q_j = (p_j^x, q_j'^y)$ or $q_j = (q_j'^x, p_j^y)$

Let ln_1 and ln_2 be horizontal and vertical lines respectively such that they cross at point h_i. The two points p_j and q_j are separated by either ln_1 or ln_2 or both. Due to symmetry, without loss of generality lets assume they are separated by ln_1 (the horizontal line through h_i).

Let's assume there exists another area A_k such that p_k is also higher than A_i. With the reasoning as for A_j, the point p_k is maximal in P; there exists q_k in A_k such that $q_k \not> A_i$ and $q_k < p_k$.

Further let's assume p_k and q_k are also separated by ln_1. Then either $p_k^x \geq p_j^x \geq q_j^x$ or $p_j^x \geq p_k^x \geq q_k^x$. As p_k and p_j lie on or above ln_1 and q_k and q_j lie below ln_1 we have that either $p_k > q_j$ or $p_j > q_k$. So one of the areas A_j or A_k must contain a point that is dominated by a point in P from another area

while its precise location point is maximal in P. Hence that area would have been updated by step 1 as by Lemma 3.

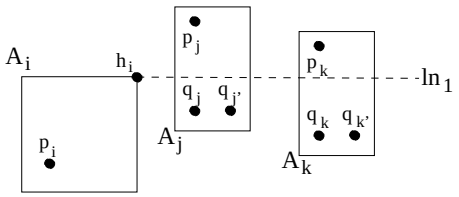

Fig. 6. Example of D-N area A_i. Step 1 of the algorithm would update area A_j

So q_k and p_k cannot be separated by ln_1 and must therefore be separated by ln_2. Following the same argument, there cannot be any further areas with precise location above h_i.

Hence there exist at most two areas with precise location higher than A_i, one separated by a horizontal line crossing h_i and another separated by a vertical line crossing h_i. The figure below shows the formation.

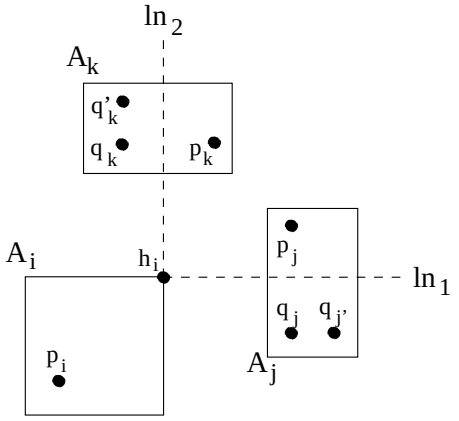

Fig. 7. D-N area A_i after step 1 of the algorithm where $p_j \in A_j$ and $p_k \in A_k$ such that $p_j, p_k > A_i$

□

As an extension of Lemma 5 we also note the following property for the case when there are two areas with precise location higher than A_i, which will be essential in a later step in phase 2:

Remark 2. If after step 1, A_i is a D-N area such that there does not exist another D-N area with precise location higher than A_i, and there exist two areas with

precise location higher than A_i then these two areas are neighbouring areas with precise locations maximal in P. In other words when all maximal areas are sorted by either x or y coordinate value of their precise location, one will be found right after the other.

We separate the cases where there is one such area (step 2) and where there are two such areas (step 3).

Step 2. Let A_i be a D-N area such that there exists exactly one area A_j with precise location p_j such that $p_j > A_i$. As A_i is a dependent area then A_j must contain at least one point that is not greater than all points in A_i. As p_j is the only point in P that is greater than A_i, without updating neither A_i nor A_j the area A_j remains dependent among $\mathcal{A}$. So any update solution must contain at least either A_i or A_j. **The action the algorithm takes here is to update** A_j, and therefore set $\mathcal{A}$ is now modified with the updated data.

At this point we give the definition of a choice that is recorded by steps 3,4 and 5 of the algorithm:

Definition 5.

A choice C_i is an update decision that has to be made between two sets of areas.

A set of updates S satisfies a choice C_i if S contains at least all the areas found in either of the two sets of C_i.

There are two types of choices: type A and type B. In type A both sets of areas are of size one, whereas in type B one set is of size one and the other is of size two.

Step 3. Let A_i be a dependent area such that there exist exactly two areas A_j and A_k with $p_j > A_i$ and $p_k > A_i$. Similar to the arguments in step 2 every update solution must contain at least one of the three areas A_i, A_j and A_k. At this stage no updates are simulated but instead **the algorithm records the choice between A_j and A_k** (type A choice).

We will use the following two Lemmas to justify the actions taken for step 2 and 3:

Lemma 6. *Let U be an update solution with $A_i \in U$. If A_i is non-maximal among $\mathcal{A}$ after the updates of $U - \{A_i\}$ then $U - \{A_i\}$ is also an update solution.*

Proof. Assume $U - \{A_i\}$ is not an update solution. So there exists an area A_j that, after the updates of $U - \{A_i\}$ are performed, is neither maximal nor non-maximal among $\mathcal{A}$. However after further updating A_i, the area A_j must become either maximal or non-maximal, as U is an update solution. Area A_j cannot be the same area as A_i since the updates of $U - \{A_i\}$ make A_i non-maximal and make A_j neither maximal nor non-maximal among $\mathcal{A}$.

If the further update of A_i changes A_j to maximal among $\mathcal{A}$ then there must have been at least one point in A_i that was higher than a point in A_j before the update. This point in A_j is non maximal as area A_i is non-maximal among $\mathcal{A}$ and any point lower must also be non maximal. This is a contradiction so A_j cannot change to maximal among $\mathcal{A}$.

If the further update of A_i changes A_j to non-maximal among $\mathcal{A}$, then point p_i is higher than all points in A_j. As A_i is non-maximal among $\mathcal{A}$ after the updates of $U - \{A_i\}$, the point p_i is also non maximal and since p_i is higher than any point in A_j it follows that A_j must have already been non-maximal among $\mathcal{A}$ before updating A_i. This is a contradiction so A_j cannot change to non-maximal among $\mathcal{A}$.

□

Lemma 7. *Let A_i be a dependent area and A_j be another area such that p_j is higher than every point in A_i. If there exists a set of areas U that is an update solution with $A_i \in U$ then $(U \bigcup \{A_j\}) - \{A_i\}$ is also an update solution.*

Proof. Since U is an update solution the set $U \bigcup \{A_j\}$ is also an update solution. As $A_i \in U$, it follows that $A_i \in U \bigcup \{A_j\}$. Since point p_j is higher than any point in area A_i and $p_j \in A_j$, after updating area A_j, the area A_i will change status to non-maximal among $\mathcal{A}$. This is also the case after updating $(U \bigcup \{A_j\}) - \{A_i\}$ since $A_j \in (U \bigcup \{A_j\}) - \{A_i\}$. Finally since $U \bigcup \{A_j\}$ is an update solution, $A_i \in U \bigcup \{A_j\}$ and A_i is non-maximal among $\mathcal{A}$ after updating $(U \bigcup \{A_j\}) - \{A_i\}$, by Lemma 6, $(U \bigcup \{A_j\}) - \{A_i\}$ is also an update solution.

□

For step 2, as by Lemma 7, every update solution containing A_i can be modified by replacing A_i with A_j and the set resulted will still be an update solution. The size does not increase since we are considering replacing only one element with another and so the update of A_j is not in every update solution but there must be an update solution of minimal size that includes A_j. For step 3, as by Lemma 7, every update solution containing A_i can be modified by replacing A_i with either A_j or A_k and the set resulted will still be an update solution. The size again does not increase and so there must be an optimal update solution that contains either A_j or A_k. These bring us to the following conclusions for step 2 and 3:

Remark 3. There exists an optimal update solution that contains the updates of step 2 and satisfies all choices of step 3.

Lemma 8. *After the updates of step 2 and the updates of a set that satisfies all recorded choices of step 3, a D-N area does not exist.*

Proof. Let's assume there exists an area A_i such that it is D-N after the updates of step 2 and after a set of updates that satisfies the choices of step 3.

If A_i contains a point that is maximal among P then $\mathcal{A} - A_i$ would not be an update solution and therefore, by Lemma 3, updated in step 1 and no longer contain that point.

So it does not contain a point maximal in P, and we now have two cases: either there does not exist another D-N area with precise location higher than every point of A_i (case 1) or there exists one (case 2).

Case 1: A by Lemma 5 there exists either one or two areas with precise location higher than A_i. If there is one then step 2 would update an area A_j

such that $p_j > A_i$ and therefore A_i would change status to N-N. So there must exist two areas with precise location higher than A_i. As the choices of step 3 are satisfied then again an update must have been made on at least one of the two areas with precise location higher than A_i, which again changes the status of A_i to N-N. So A_i cannot be dependent, contradiction.

Case 2: There must exist some other D-N area which falls to case 1 and has precise location higher than A_i. As the other D-N area falls in case 1 it would have change status to N-N. Therefore this case will also be managed indirectly as any point lower than a non-maximal point is also non-maximal. So A_i cannot be dependent, contradiction.

□

Steps 4 and 5.

In the fourth and fifth steps the remaining partly areas with maximal precise location (P-M) are considered. After step 3 for each P-M area there exist either one or two areas interfering with it, formally:

Lemma 9. *Let A_i be a P-M area with precise location p_i. Further let $\mathcal{R}$ be the set of all areas interfering with A_i. Set $\mathcal{R}$ contains either one or two areas.*

Proof. Recall the restriction proposed earlier for the uncertainty areas. As by the restriction area A_i is made by the direct product of a set A_i^x and a set A_i^y.

Let $l_i = (l^x, l^y)$ where l^x is the lower limit of A_i^x and l^y is the lower limit of A_i^y. We note the point l_i may or may not be in A_i and it is dominated by all other points in A_i. Formally if $l_i \in A_i$ then $q_i \geq l_i$ for every point $q_i \in A_i$, and if $l_i \notin A_i$ then $q_i > l_i$ for every point $q_i \in A_i$.

Furthermore each point in A_i is maximal among P as otherwise the instance cannot be solved without updating A_i and hence A_i would have been updated in step 1.

Also as A_i is partly there must exist an area A_j such that it contains a point q'_j with $q'_j > q_i$ for some $q_i \in A_i$, and has the precise location $p_j \not> q_i$ as each point in A_i is maximal among P. We now choose a point $q_j \in A_j$ with either $q_j = (q_j'^x, p_j^y)$ or $q_j = (p_j^x, q_j'^y)$ such that $q_j > q_i$ and $q_j > p_j$ (see figure below). As A_j is a product of possible x and y values point q_j exists in A_j. We note that also $q_j > l_i$ as $q_i \geq l_i$. The figure below shows these properties.

Fig. 8. P-M area A_i with $q_j = (p_j^x, q_j'^y)$ or $q_j = (q_j'^x, p_j^y)$

Area A_j cannot be N-N as it contains a point higher than a point in A_i. If A_j was N-N then there must have been a point in P higher than A_j, which in

turn would also be higher than a point in A_i and contradict the fact that each point in A_i is maximal among P. Furthermore by Lemma 8 it cannot be D-N after the updates of step 2 and the set of updates that satisfy the choices of step 3. Also A_j cannot be D-M or P-N as by Lemma 4 after the updates of step 1 there are no D-M and P-N areas left. So A_j is either a M-M or P-M area. In all cases the point p_j is maximal in P.

In both cases $l_i \in A_i$ and $l_i \notin A_i$ for every point $q_i \in A_i$ we have that $p_j \not\succ q_i$, as A_i only contains points that are maximal among P.

Case $l_i \in A_i$: As $p_j \not\succ q_i$ for all $q_i \in A_i$ then also $p_j \not\succ l_i$.

Case $l_i \notin A_i$: As l_i is made by the lower limits of A_i^x and A_i^y, either the lower limit of A_i^x is not in A_i^x or the lower limit of A_i^y is not in A_i^y, or both limits are not in their sets. As $p_j \not\succ q_i$ and $q_i > l_i$ for all $q_i \in A_i$, if $p_j^x > l_i^x$ then $p_j^y \leq l_i^y$ and if $p_j^y > l_i^y$ then $p_j^x \leq l_i^x$. Otherwise $p_j \not\succ l_i$.

Let ln_1 and ln_2 be horizontal and vertical lines respectively such that they cross at point l_i. The two points p_j and q_j are separated by either ln_1 or ln_2. Due to symmetry, without loss of generality lets assume they are separated by ln_1 (the horizontal line through l_i).

Let's assume there exists another area A_k such that it contains a point q_k with $q_k > q_i$ for some $q_i \in A_i$. With the reasoning as for A_j, we have $q_k > p_k$ and $p_k \not\succ q_i$ for all $q_i \in A_i$.

Further let's assume p_k and q_k are also separated by ln_1. Then either $p_k^x \geq q_j^x \geq p_j^x$ or $p_j^x \geq q_k^x \geq p_k^x$. As p_k, p_j lie below ln_1 (or on ln_1 for case $l_i \notin A_i$) and q_k, q_j lie above ln_1 (or on ln_1 for case $l_i \in A_i$) we have that either $q_k > p_j$ or $q_j > p_k$. So although one of the areas A_j or A_k has the precise location point maximal in P, without updating the area that interferes with this precise location there cannot be an update solution. Hence that area would have been updated by step 1.

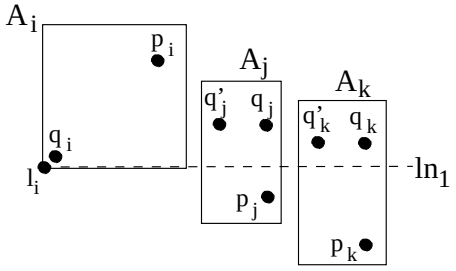

Fig. 9. Example of P-M area A_i. Step 1 of the algorithm would update area A_j

So q_k and p_k cannot be separated by ln_1 and must therefore be separated by ln_2. Following the same argument, there cannot be any further areas containing a point higher than a point in A_i.

Hence there exist at most two areas with precise location higher than A_i, one separated by a horizontal line crossing h_i and another separated by a vertical line crossing h_i.

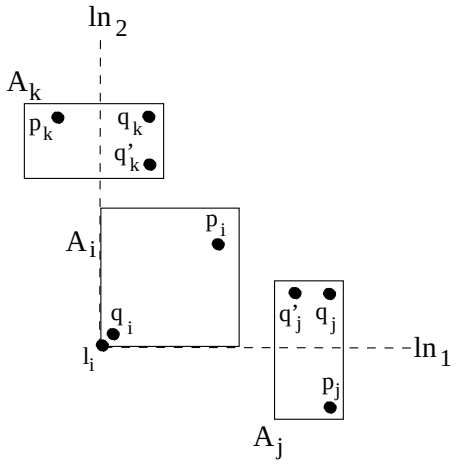

Fig. 10. P-M area A_i after step 1 of the algorithm where $q_j \in A_j$ and $q_k \in A_k$ such that $q_j, q_k > q_i$

□

By Lemma 9, at this point, a partly area A_i with maximal point can have either one or two areas interfering with it, and the precise locations of which are not higher than any point in A_i. Again we separate these cases accordingly (Step 4 for one area and Step 5 for two areas.)

Step 4. Let A_i be a partly area such that there exists exactly one area, namely A_j, interfering with A_i. As the precise location of A_j is not higher than any point in A_i, updating A_j will change the status of A_i to maximal among $\mathcal{A}$. Furthermore by updating A_i this is also the case as there is no point higher than the precise location of A_i. As A_j contains at least one point higher than a point in A_i, without updating neither A_i nor A_j, area A_i will remain partly and hence any update solution must contain at least one of the two. At this stage an update cannot be simulated and therefore the **algorithm records the choice between A_i and A_j** (type A choice).

Step 5. We now consider two areas namely A_j and A_k which interfere with area A_i. With a similar reasoning as in step 4 we have that without updating neither A_i nor both A_j and A_k area A_i remains partly and so every update solution must contain at least either A_i or both A_j and A_k. Again an update will not be simulated and therefore the **algorithm records the choice between area A_i and the set of areas consisting of A_j and A_k** (type B choice).

With the collection of all choices of steps 4 and 5 we have the following:

Remark 4. Every update solution must satisfy all choices collected by step 4 and step 5.

Lemma 10. *Any set of updates that satisfies all choices of step 4 and step 5 does not leave any partly areas of maximal points (P-M).*

Proof. Let's assume there exists an area A_i such that it is P-M after satisfying the choices of steps 4 and 5.

As A_i is partly among $\mathcal{A}$ it contains at least two points and there exists an area A_j with a point $q_j \in A_j$ such that q_j is higher than a point $q_i \in A_i$. Also as p_i is maximal among P then $p_j \not> p_i$.

We have that either $p_j > q_i$ (case 1) or $p_j \not> q_i$ (case 2).

Case 1: If this is the case then $\mathcal{A} - A_i$ is not an update solution and therefore A_i would have been updated in step 1 and it would have trivial size. As $p_i, q_i \in A_i$ and $p_i \neq q_i$ this is a contradiction.

Case 2: In this case as by steps 4 and 5 either A_i itself must be updated to satisfy the choice or all areas interfering with A_i must be updated. As A_i is partly and not trivial then all areas interfering with A_i must have been updated. Since $q_j > q_i$ and $p_j \not> q_i$ then $p_j \neq q_j$. As $p_j, q_j \in A_j$ and $p_j \neq q_j$ then A_j was not updated and therefore the choice was not satisfied, contradiction.

□

We also note by the following Lemma 11 for P-M areas, a property that is essential in phase 2.

Lemma 11. *Let A_i be a partly area and let set R contain the areas interfering with A_i. Then A_i and the areas in $\mathcal{R}$ are neighbouring areas, with A_i being in the middle if $\mathcal{R}$ is of size 2.*

Proof. Let A_j and A_k be the areas in $\mathcal{R}$, with p_i and p_j being the precise locations of A_j and A_k respectively. Further let area A_j be located to the top left side of p_i and A_k be located to the bottom right side of p_i.

Finally let $lh1$ and $lv1$ be horizontal and vertical lines respectively intersecting on point p_i, and similarly let lines $lh2$ and $lv2$ intersect on p_j. Then $lh1$ must also intersect with $lv2$, and $lv1$ with $lh2$. The resulting rectangle forms the region between A_i and A_j, where no other area can have a point into except on lines $lv2$ and $lh1$.

If there was an area A_p which had every point higher than the point of $lv1 \cap lh2$, and still had a point in the rectangle, then at least one of the following scenarios would occur which the algorithm would have managed in a previous step: A_p would be P-M or D-M such that $\mathcal{A} - A_p$ is not an update solution and therefore updated in step 1, or A_i would be dependent instead of partly among $\mathcal{A}$.

If the whole area A_p was not higher than the point of $lv1 \cap lh2$ but still had a point higher than $lv1 \cap lh2$, and not on the lines of $lv1$ and $lh2$, then A_p would be self determined and the algorithm would have updated it in step 1.

This proves that no area can be found between A_i and A_j.

In a similar way a rectangle can be constructed between p_i and p_k and it can be shown that no area can be found between A_i and A_k.

Since there is no area found between A_i and A_j and no area found between A_i and A_k it follows that A_i can only be in the middle of A_j and A_k. □

Having analysed the steps of phase 1 we now combine the results to show the desired outcome:

Lemma 12. *a) All updates made in phase 1 together with a set of updates that satisfies the choices collected is an update solution.*

b) All updates made in phase 1 together with a minimal set of updates that satisfies the choices collected is an optimal update solution.

c) Phase 1 runs in polynomial time.

Proof. a) It was shown after updating various areas and satisfying the collected choices how each of the types D-M/D-N/P-M/P-N is eliminated, therefore leaving only the types M-M and N-N as needed to have an update solution.

b) Furthermore at actions which perform updates it has been shown that either the updates must be in every update solution (step 1) (use of Lemma 3) or the updates must be in at least one optimal update solution (step 2) (use of Lemmas 6 and 7). Hence the set of updates so far are minimal. It is also shown that if one further satisfies all the collected choices (from steps 3,4,5) (use of Lemmas) with another minimal set of updates then the two sets together form an optimal update solution.

Moreover if there is a set of updates that does not satisfy all choices, then it is not an update solution of minimal size, as explained in steps 3,4,5 and by the definition of choices.

c) We now show the time complexity for the various steps.

Classifying all areas to the types D-M/D-N/P-M/P-N takes at most n^2 to compare each area against all others. This needs to be done before the execution of the algorithm, and also after each step that performs updates, specifically after steps 1 and 2 of phase 1.

For step 1 the algorithm needs to compare each area against all others and find out if Lemma 3 holds and perform updates if that is the case. There is n number of areas and therefore this takes at most n^2 times.

For Steps 2 and 3 the algorithm needs to find the one or two areas which interfere with each dependent area, as by Lemma 5. There are at most n dependent areas and therefore we have at most n^2 iterations to find the areas that interfere with the dependent areas.

For steps 4 and 5 the algorithm needs to find the one or two areas which interfere with each partly area, as by Lemma 9. There are at most n partly areas and therefore we have at most n^2 iterations to find the areas that interfere with the partly areas.

After this phase 1 finishes and therefore it runs in polynomial time.

□

3.2 Phase 2

After the completion of phase 1 a set of updates and a of set choices have been established. The set of updates so far is minimal and it remains to further choose a minimal number of updates to satisfy all the recorded choices, in order to provide an optimal update solution. Recall that the choices only contain areas with precise location maximal among P. These choices are then split into chains and all areas within the choices are sorted top-left to bottom-right. This can be done just in the order of increasing x. The properties of choices and chains are as follows:

Remark 5.

Steps 3 and 4 of the algorithm produce type A choices.

Step 5 of the algorithm produces type B choices.

As stated earlier, choosing neither of the two update sets of a choice does not produce an update solution. Furthermore choosing either of the two update sets of a choice is enough to satisfy the choice.

Definition 6. *A choice C_1 overlaps with a choice C_2 if there exists an area A such that $A \in C_1$ and $A \in C_2$.*

Definition 7. *A chain is a sequence of choices ($C_1, \ldots, C_k$) where C_i overlaps with C_{i+1} for all $1 \leq i \leq k-1$ and the sequence is of maximal length.*

We have also establish in Lemmas 2 and 11 that the choices are between neighbours. That is for type A a choice has to be made between 2 areas that are next to each other, and for type B a choice has to be made between the middle area and the 2 side areas. An example of a chain with the different kind of choices is show below.

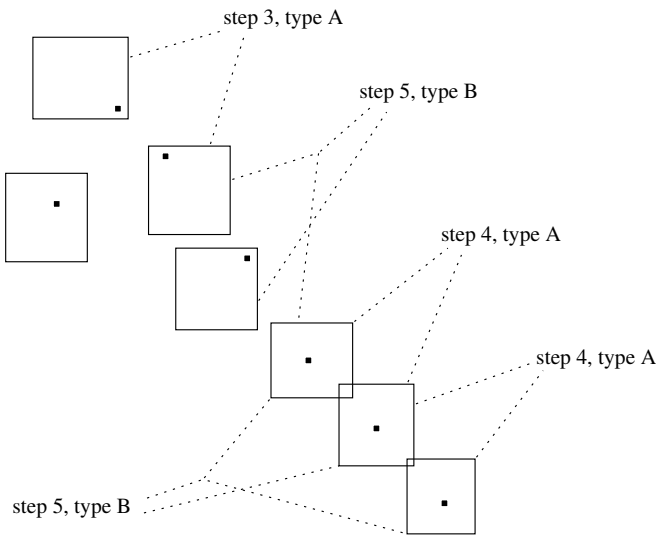

Fig. 11. Example of a chain of choices

By sorting the areas in the chains the following Lemma takes place:

Lemma 13. *By choosing the second and every other area in a chain, all choices in the chain are satisfied. Furthermore no two consecutive areas in a chain can be left out of an update solution.*

Proof.

By choosing all even areas in the chain, for every choice in the chain either every odd or every even area is chosen.

For a choice of type A choosing the odd or the even area satisfies the choice.

Hence by choosing each even area in a chain all choices of the chain are satisfied.

By Definition 7 two consecutive choices in a chain are overlapping.

If they form a type A choice then they cannot both be left out of an update solution.

If they are part of a type B choice then we have that the middle area is the one element of the choice and the leftmost and rightmost areas form the other element of the choice. Hence by leaving out either the leftmost and middle or the middle and rightmost areas no valid element of choice remains and therefore they cannot both be left out of an update solution.

□

As by Lemma 13, by updating every second area in every chain all choices are satisfied and when combined with the simulated updates of phase 1 an optimal update solution is produced. Lemma 12 further shows the time complexity of the algorithm.

The time complexity in phase 2 is follows. To sort the areas a basic sorting algorithm will take $\mathcal{O}(n^2)$. Each choice is then compared against the next one to decide if the next one belongs in the same chain or a new chain needs to be created. This will take at most n iterations, in the case there is n number of areas in total for all choices. Satisfying the chains of choices and returning the collection of every second area in a every chain takes at most $n/2$ iterations.

As all phases of the algorithm run in polynomial time and the output is an optimal update solution theorem 1 is satisfied.

References

1. Susanne Albers and Pascal Weil, editors. *STACS 2008, 25th Annual Symposium on Theoretical Aspects of Computer Science, Bordeaux, France, February 21-23, 2008, Proceedings*, volume 1 of *LIPIcs*. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, Germany, 2008.
2. Richard Bruce, Michael Hoffmann, Danny Krizanc, and Rajeev Raman. Efficient update strategies for geometric computing with uncertainty. *Theory of Computing Systems*, 38(4):411–423, 2005.
3. George Charalambous and Michael Hoffmann. Verification problem of maximal points under uncertainty. In Thierry Lecroq and Laurent Mouchard, editors, *IWOCA*, volume 8288 of *Lecture Notes in Computer Science*, pages 94–105. Springer, 2013.
4. Thomas Erlebach and Michael Hoffmann. Minimum spanning tree verification under uncertainty. 2014.
5. Thomas Erlebach, Michael Hoffmann, Danny Krizanc, Matús Mihalák, and Rajeev Raman. Computing minimum spanning trees with uncertainty. In Albers and Weil [1], pages 277–288.
6. T. Feder, R. Motwani, R. Panigrahy, C. Olston, and J. Widom. Computing the median with uncertainty. *SIAM Journal on Computing*, 32(2):538–547, 2003.
7. Manoj Gupta, Yogish Sabharwal, and Sandeep Sen. The update complexity of selection and related problems. In *IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS 2011)*, volume 13 of *LIPIcs*, pages 325–338. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2011.
8. Simon Kahan. A model for data in motion. In *Proceedings of the 23rd Annual ACM Symposium on Theory of Computing (STOC'91)*, pages 267–277, 1991.