ONLINE AND VERIFICATION PROBLEMS UNDER UNCERTAINTY

Thesis submitted for the degree of
Doctor of Philosophy
at the University of Leicester

by

George Charalambous
Department of Computer Science
University of Leicester

August 2015

Abstract

In the *under uncertainty* setting we study problems with imprecise input data for which precise data can be obtained. There exists an underlying problem with a feasible solution, but is computable only if the input is precise enough. We are interested in measuring how much of the imprecise input data has to be *updated* in order to be precise enough. We look at the problem for both the online and the offline (verification) cases. In the *verification under uncertainty* setting an algorithm is given imprecise input data and also an assumed set of precise input data. The aim of the algorithm is to update the smallest number of input data such that if the updated input data is the same as the corresponding assumed input data (i.e. verified), a solution for the underlying problem can be calculated. In the *online adaptive under uncertainty* setting the task is similar except the assumed set of precise data is not given to the algorithm, and the performance of the algorithm is measured by comparing the number of input data that have been updated against the result obtained in the verification setting of the same problem.

We have studied these settings for a few geometric and graph problems and found interesting results. Geometric problems include several variations of the maximal points problem where, in its classical form, given a set of points in the plane we want to compute the set of all points that are maximal. The uncertain element here is the actual location of each point. Graph problems include a few variations of the graph diameter problem where, in its classical form, given a graph we want to calculate a farthest pair of vertices. The uncertain element is the weight of each edge.

Acknowledgements

First, I would like to thank my supervisor Dr. Michael Hoffmann for all his invaluable help throughout the four years of my research and the completion of my thesis.

I am also grateful to my second supervisor Prof. Thomas Erlebach for all his support and advice. I would also like to thank Dr. Fer-Jan De Vries for assessing my work at the end of each year.

Special thanks to the University of Leicester and the Department of Computer Science for the financial support of my studies and my attendance in conferences in Europe.

Finally I would also like to thank all the colleagues and members of stuff in the Department of Computer Science and all my friends in Leicester that formed a friendly environment during all these years of my studies.

Contents

Chapter 1

Introduction

1.1 Motivation

Most classical algorithms produce an output with an assumed precise input, for example the sorting of n numbers. However, this assumption is untrue in many scenarios. For example the data on which algorithms run is increasingly being obtained automatically, or by automatic pre-processing of raw data. Such data has many uncertainties. Some reasons are that sensors have errors, data may be out-of-date or missing and may have to be "guessed" by inference procedures.

There are many approaches to dealing with uncertainty. For example one may model data probabilistically, or allow the algorithm to produce only approximate answers. Of course one can get uncertain answers given uncertain data, however, can we get exact answers to problems when the data is uncertain? The approach taken in this thesis is somewhat different from the majority of other approaches on dealing with uncertainty. The main assumption is that although data may be uncertain, the *precise value of any particular datum may be obtained* by an *update* operation, but this comes at a (significant) cost. At any point an algorithm may have some precise data while at the same time some uncertain data. The algorithm is then asked to act based on the current, possibly uncertain, information and if the current information is insufficient then updates must be considered. So the task of the algorithm is to request updates until the information allows for a provably correct solution of the underlying problem to be calculated. Of course the algorithm can always provide such a solution by updating all of the data, but this is assumed to be too expensive. Generally, minimizing the use of updates is desired. Problems *under uncertainty* capture this setting which is the setting considered in this thesis.

Precision may be time, energy and bandwidth consuming, require large computational power or for other reasons even infeasible. Accurate and timely stock quotes cost money. Remote access to the state of network queues costs time and bandwidth.

Querying battery-powered units of sensor networks unnecessarily uses up precious energy. 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. With GPS sensors in mind, the current measurement may be sent automatically if this exceeds some predefined bounds of the latest sent one. Hence, based on the last sent information, a possible band or area is known where the current measurement of the sensor is within.

We refer to such uncertain data as *uncertainty areas* or *areas of uncertainty.* We say the set of all uncertainty areas that have been updated after an algorithm has finished its execution is an *update solution* and if this set is also of minimal size (i.e. there is no algorithm that can make less updates even by having knowledge of what the updates produce) then we say this is an *optimal update solution.* We may also call the algorithm an *update strategy.*

1.2 Thesis Overview

In Chapter 2 we give the background and preliminaries of problems under uncertainty in general. This includes the various categories and evaluation measures, formal definitions and a general solving method for this type of problems. In Chapter 3 we give a survey of known results in our field. In Chapter 4 we give the definitions and general notation for the maximal points problem, which we use throughout the following chapters (Chapter 5, 6, 7 and 8) where we study various different aspects of the problem.

Our results on Maximal Points. In Chapter 5 we look at the verification setting of the problem and show that it is NP-Hard if there are no restrictions in place for the accepted areas of uncertainty. This work has been published in [9]. In Chapter 6 we propose a new model for the accepted input and we give an update strategy for the verification setting that runs in polynomial time whilst for the online adaptive setting the 3-update competitive strategy by Bruce et al. [6] still applies. For this new model, in Chapter 7, we also give a strategy with optimal competitive ratio for the slightly different problem where one can perform coordinate-specific updates. Finally in Chapter 8 we look at the problem in dimensions higher than 2D, showing specifically that in 3-dimensional space there is no update strategy with constant competitive ratio.

Our results on Farthest Pair of Vertices. In Chapter 9 we study the problem of finding a farthest pair of vertices in graphs with uncertain edge weights. In Section 9.3 we show for trees matching lower and upper bounds if the input consists of only combinations of open intervals or singletons, and if closed intervals are allowed we show that there is no update strategy with constant competitive ratio. We further show in Section 9.4 the lower bound for cycle graphs.

Finally in Chapter 10 we give a few ideas for possible directions of future work.

Chapter 2

Preliminaries

2.1 Categories and Evaluation Measures

Work in computing under uncertainty falls in three main categories/settings: *adaptive online*, *non-adaptive online* and *verification*.

In the adaptive online setting an algorithm initially knows only the uncertainty areas 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. In the non-adaptive online setting an algorithm is again given only the uncertainty areas initially, but it must determine a set of updates such that after performing all updates in the set, it is guaranteed to have sufficient information to determine a solution.

Usually, to assess the performance of an algorithm in the adaptive / non-adaptive online setting, we compare the number of updates that the algorithm makes to the optimal number of updates which we denote as OPT; which follows the common approach of competitive analysis for online algorithms [4]. Here optimality is defined in terms of a prescient adversary who, knowing the values of all input parameters, makes the fewest updates needed to present a solution to the problem that is provably correct. In other words, after the updates claimed by the adversary are made, no additional uncertainty areas need to be updated to verify the correctness of the solution, and this could not have been achieved with less updates. We say that an algorithm is *k-update competitive* or has a *competitive ratio* of k if, for each input instance, the algorithm makes at most k times as many updates as OPT for that input instance. Formally, an algorithm is k-update competitive if it makes at most $k * OPT + \mathcal{O}(1)$ updates for every instance of the problem, where $\mathcal{O}(1)$ is a constant independent of the instance.

When studying online problems under uncertainty, bounds can be constructed that reason about what is the best possible by any online algorithm and what is the worst case for a specified online algorithm. In this context, we call the former the *lower bound* and the latter the *upper bound*. The upper bound is another term to what we have already mentioned earlier as competitive ratio. Similarly, the lower bound is the performance ratio against OPT that can achieved, but by any possible online algorithm. For an online problem under uncertainty, it is ideal to construct an algorithm that solves the problem with upper bound that matches the problem's lower bound. If this is the case then we say the algorithm achieves an *optimal competitive ratio*.

In the verification setting an 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 all updates are verified. It is worth noting that the solution size of the verification setting for a problem under uncertainty is essentially OPT, which can be used for the competitive analysis of that problem in the adaptive/non-adaptive online setting.

For a verification under uncertainty problem usually we aim to construct an algorithm that solves the problem in polynomial time, or failing that, to show that the problem is NP-hard.

Note that there are no results in this thesis that fall under the non-adaptive online setting. We give formal definitions in the next section.

2.2 Definition of a Problem Under Uncertainty

We now give the formal definitions of a problem under uncertainty and a witness algorithm, as were also given in [16, 9, 29].

An instance (S, U, A, w) of a problem under uncertainty consists of the following components. S represents some structural information and U is a set of elements with uncertain values. For each element $u \in U$ the function A maps u to an uncertainty area A_u and the function w maps u to its precise value w_u. Note that $w_u \in A_u$ is required and depending on the nature of the problem considered w_u may be a real number, coordinates in the plane or any other type of input data. An algorithm is able to update u which can be seen as the operation of replacing the contents of A_u with the singleton set $\{w_u\}$. After this we also say A_u is *trivial*.

For a given instance $I = (S, U, A, w)$ we let $\phi(S, U, w)$ denote the set of all solutions for I. The input of an algorithm in the adaptive/non-adaptive online setting is (S, U, A) and the task is to perform enough updates such that an element of $\phi(S, U, w)$ is computable (or such that all of them are computable depending on the variation of the problem). The output is the set containing all the updates performed during its execution which we call, as mentioned in Section 1.1, an update solution. After updating all elements of U then obviously the output is an update solution but we aim to minimize the use of updates and evaluate the performance of an algorithm as per the previous section. On the other hand the input of an algorithm in the verification setting is (S, U, A, w) and the output set additionally has to be minimal, or in other words the algorithm has to produce an optimal update solution. Finally we say I is a *solved instance* if $\emptyset$ is an update solution for I.

For example definitions see maximal points in Chapter 4 and farthest pair of vertices in Section 9.2.

2.3 A General Observation

If for a problem under uncertainty the task of an online algorithm is to find just a single update solution, which is usually the case, we often observe the following behaviour. There is no algorithm with constant competitive ratio under the assumption that closed intervals are allowed as uncertainty areas; whereas if each uncertainty area is guaranteed to be either trivial or an open interval, there exists an update competitive algorithm. Examples of problems where this is observed include the minimum spanning tree by Erlebach et al. [29] and our farthest pair of vertices in Chapter 9.

The difficulty with closed intervals appears where situations can exist such that multiple non-trivial uncertainty areas share the same upper (or lower) limit l, and there exists one of them which, when updated, is actually reduced to $\{l\}$. If this single update can prove the corresponding element is maximum (or minimum) among the candidates considered, then it might be enough to solve the problem. So the prescient adversary only chooses such an update, but no online update strategy has a way of determining which one it is and therefore can be forced to make a large number updates.

If an online algorithm is instead asked to find all update solutions, then a constant competitive ratio is often achievable also for when closed intervals are allowed as uncertainty areas. An example for this is the problem of finding the maximum, the median, and the minimal gap between any two numbers by Kahan [32].

Furthermore if the task is to compute the lexicographically smallest solution a constant competitive ratio might also be possible for closed intervals, as shown for some cases of the minimum spanning tree and selection problems by Gupta et al. [25]. This notion is a natural version in many problems where the initial ordering is important and it was shown in [25] that this has the desired effect of limiting non-deterministic guessing powers of OPT.

2.4 The Witness Algorithm

Usually, online update strategies that handle problems under uncertainty are based on a general method called the *witness algorithm*. The witness algorithm was first introduced by Bruce et al. [6] and described in a more general setting along with some of its properties by Erlebach et al. [29]. Using this method, a witness set W is a set of uncertainty areas such that, in order to verify a solution, at least one element of W needs to be updated. The algorithm updates all areas in W and if the problem is not solved then repeats the process to construct and update the next W. Assuming then that at any state either the problem is solved or a set W can be found, the algorithm is k-update competitive where k is the maximum size of any witness set W. Following the above we remark:

Remark 2.1. *If a witness algorithm computes t witness sets during its execution, then $OPT \geq t$.*

For an instance $I = (S, U, A, w)$ of a problem under uncertainty with $\phi(S, U, w)$ denoting the set of all solutions for I, the set $W \subseteq \bigcup_{u \in \mathcal{U}} A_u$ is a witness set if no element of $\phi(S, U, w)$ can be verified without updating an element of W. The general witness algorithm is shown below:

Algorithm 1 The general witness algorithm

```
  Initialize: Input $(S, U, A)$
  while An element of $\phi(S, U, w)$ cannot be verified do
      Find a witness set $W$ ; Update all elements of $W$;
      Input changes to $(S', U', A)$;
  end while
      return The set of all uncertainty areas that have been updated.
```

Note that the above algorithm is not consistent with the non-adaptive online setting. The difference is that an algorithm does not get to make multiple iterations but instead must determine an update solution in a single iteration, as all updates must be made in parallel.

As mentioned in the beginning of this section, most of the results in the literature regarding online algorithms with constant competitive ratio for problems under uncertainty are based on the template of the witness algorithm. Furthermore the competitive ratio is directly linked with the maximum size of witness sets, in other words:

Lemma 2.2. *If the size of every witness set used by a witness algorithm is bounded by k, then the witness algorithm is k-update competitive.*

Lemma 2.2 can be proved, for a particular problem considered, by showing that every update solution must contain from each computed witness set a distinct element.

Chapter 3

Related Work

3.1 Under Uncertainty Problems

Kahan [32] 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 (or *lucky ratio* as he has defined it).

Feder et al. [19] also studied the problem of computing the value of the median of an uncertain set of numbers up to a certain tolerance. Each input number lies in an interval and an update reveals the exact value, but different intervals have different update costs. They consider offline algorithms, which must decide the sequence of updates prior to seeing the answers, as well as online ones, aiming to minimize the total update cost.

Olston and Widom [44] consider selection and aggregation problems in the non-adaptive online case, motivated by the situation where one maintains in a local cache intervals containing the actual data values stored at a remote location. For a given tolerance δ, the aim is to determine a range $[L, H]$ which contains the exact solution such that $H - L \leq \delta$.

Khanna and Tan [35] extend some of the results for the selection and sum problems, focusing on other precision parameter formulations.

In the setting of geometric problems with points where their location in the plane is imprecise, Bruce et al. [6] studied the problems of computing maximal points and the convex hull of a set of points. Here, the input consists of 2-dimensional points in the plane, and the uncertainty information is for each point of the input an area that contains that point. They presented update strategies for both problems with competitive ratio of 3 and shown that this is the best possible. They introduced a general method, called the *witness algorithm*, for dealing with problems involving uncertain data and derived their update strategies using that method. Chapter 5, 6, 7 and 8 in this thesis further analyse other aspects and different settings of the maximal point problem.

Löffler and van Kreveld [41] have studied the problem of computing the largest or smallest convex hull over all possible locations of points inside their uncertainty areas. Here, the option of updating a point does not exist, and the goal is to design fast algorithms computing an extremal solution over all possible choices of exact values of the input data.

Examples of under uncertainty applications to graphs include Feder et al. [18] where they investigated algorithms for computing the length of a shortest path from a source s to a given vertex t on graphs with uncertain edge weights. They allowed a precision factor limiting the deviation from the actual shortest path and they studied the computational complexity of minimizing the total update cost.

Erlebach et al. [29] studied the adaptive online setting for minimum spanning trees (MST) under two types of uncertainty: the edge uncertainty setting, which is the same as the one considered by Feder et al. [18], 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 problems they presented algorithms with optimal competitive ratio. The competitive ratios are 2 for edge uncertainty and 4 for vertex uncertainty, and the uncertainty areas must satisfy certain restrictions for example in the edge uncertainty case each uncertainty area must be either open or trivial.

Erlebach et al. [15] further worked on the MST problem, for both edge and vertex uncertainty, this time studying the verification setting. 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.

Megow et al. [42] also worked on the MST under uncertainty showing that randomized query strategies can beat the competitive ratio 2 of deterministic algorithms. Their randomized algorithm achieves expected competitive ratio $1 + 1\sqrt{2} \approx 1.707$. Moreover, they gave an optimal algorithm for the related problem of computing the exact weight of an MST at minimum query cost. Finally they show the results also hold for the more general setting of matroids.

Gupta et al. [25] studied a variant of the MST problem under uncertainty, along with the selection problem, where updates yield more refined estimates in the form of sub-intervals instead of precise values. They generalized the update model in several directions, classifying models based on the types of the inputs allowed and the return type of the updates. In the generalized model they present 2-update competitive algorithms that do not depend on the lengths or distribution of the sub-intervals. They further show that for models with closed intervals, update-competitive algorithms become possible if the output is required to be a lexicographically smallest solution.

Kirkpatrick [36] considered a basic list-searching problem: given a set of lists with unknown lengths, traverse to the end of any one of the lists with as little total exploration as possible. They introduced a novel process interleaving technique, called hyperbolic dovetailing that achieves a competitive ratio that is within a logarithmic factor of optimal on all inputs in the worst, average and expected cases, over all possible deterministic (and randomized) dovetailing schemes. They also show that no other dovetailing strategy can guarantee an asymptotically smaller competitive ratio for all inputs.

A problem under uncertainty where updates yield more refined estimates was also considered by Tseng and Kirkpatrick [50]. They study the complexity of one-dimensional extrema testing: given one input number, determine if it is properly contained in the interval spanned by the remaining input numbers. They assume that each number is given as a finite stream of bits, in decreasing order of significance. The performance of an algorithm is measured by the total number of bits that it needs to consume from its input streams. and an input-thrifty algorithm is one that performs favourably with respect to this measure. They present an algorithm for the extrema testing problem that is within a logarithmic factor of the intrinsic cost of the given instance. They further mention that their results also hold in a more general model where the query results are nested uncertainty intervals, instead of a query yielding an extra bit of a number.

Erlebach et al. [17] studied under uncertainty the problem of identifying a cheapest set among a given collection of feasible sets using a minimum number of updates of element weights. For the general case they present an algorithm that makes at most $kOPT + k$ updates, where k is the maximum cardinality of any given set. For the minimum multi-cut problem in trees with k terminal pairs, they give an algorithm that makes at most $kOPT + 1$ updates. For the problem of computing a minimum-weight base of a given matroid, they give an algorithm that makes at most $2OPT$ updates, generalizing the result in [29] for the minimum spanning tree problem. For each of their algorithms they further give matching lower bounds.

A survey of known results and techniques for the design of query-competitive algorithms in the model of computing under uncertainty was given in [16].

3.2 Classical Maximal Points

The problem is simple: Given a set of points, return all that are not dominated. In the literature this is also called the problem of computing the maxima (or maximum) of a set of vectors (or points). Other names include the Pareto optimum and, more recently, the skyline operator. A formal definition of the maximal points problem in the plane can be found in Chapter 4.

The problem was introduced by Kung et al. [38] and was noted in [46] that it has a variety of applications in statistics, economics, and operations research. In two and three dimensions, the best known algorithm solves the problem in $\mathcal{O}(n \log n)$ time which is optimal since the problem exhibits a sorting lower bound as shown in [46, 38]. For higher dimensions it was shown by Kung et al. [38] and Gabow et al. [23] that the problem can be solved in $\mathcal{O}(n \log^{d-2} n)$ time and $\mathcal{O}(n \log^{d-3} n \log\log n)$ time respectively.

This problem has subsequently been studied in many other contexts, including solutions for parallel computation models [31], for point sets subject to insertions and deletions [33], and for moving points [21]. Output-sensitive algorithms and algorithms that find the maxima for a random set of points are described in [1, 12, 24, 37].

The problem was first introduced to the database community in [5] as the SQL skyline operator. It has since received a lot of research attention for various settings and many different approaches to efficiently compute skyline queries with over a hundred publications in recent times. A 2012 survey of skyline processing in highly distributed environments can be found in [30]. More recent work in the area includes [2, 3] where they exploit the computational power of the GPU, [43, 10, 53] using the MapReduce platform and [39, 40, 54] for skyline queries over distributed uncertain data.

3.3 Classical Graph Diameter

Computing the diameter and, more generally, computing distances, are among the most fundamental algorithmic graph problems. A survey can be found in [55] for many different settings and models. In Chapter 9 we look at a problem under uncertainty that is directly related to the problem of computing the diameter of a graph. Specifically we are interested in finding a farthest pair of vertices in undirected and non-negatively real-weighted graphs, where the edge weights may be uncertain.

The best known algorithm for finding the diameter exactly is by running an algorithm for the all-pairs shortest paths problem (APSP) and returning the largest distance. It is a long-standing open problem whether one can compute the diameter faster than APSP [11]. APSP is unquestionably one of the most well known problems in algorithm design, frequently studied in textbooks. For arbitrary dense real-weighted graphs with n vertices, Floyd [20] and Warshall [51] proposed an algorithm for ASPS in 1962 that runs in $\mathcal{O}(n^3)$ time (for a recent publication see [13]). Over the years there has been a tremendous amount of work to improve its running time [7, 8, 14, 22, 26, 27, 28, 47, 48, 49, 52, 56]. The best known algorithm so far has been proven by Williams [52] to run in $\mathcal{O}(\frac{n^3}{2^{c\sqrt{\log n}}})$ time for some constant c. For sparse graphs an algorithm with $\mathcal{O}(mn \log \alpha(m, n))$ time for any graph with m edges has also been achieved by Pettie and Ramachandran [45].

Chapter 4

Maximal Points Introduction

In this chapter we give an introduction of the general maximal point under uncertainty problem that was first introduced by Bruce et al. [6] and further studied by us in the following chapters. We will give some definitions for the classical problem as well as for the problem under uncertainty, and we will use this terminology throughout Chapter 5, 6, 7 and 8.

In its simple form, without uncertainty, it can be stated as follows: Given a set of points in the plane, return all points that are maximal (we say a point is maximal if there does not exist some other point with one coordinate greater and all other coordinates not lower). Considering points in a plane, a point p may be written in the 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 $p > q$ and this notion naturally extends to higher dimensions. We are interested in finding all points such that there does not exist a point higher; that is find all points in a set that are *maximal*. Formally:

Definition 4.1. *Let p be a point and let P be a set of points. 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 now use the under uncertainty setting in the context of the maximal points problem, as by our definitions in Section 2.2, motivated by scenarios where the location of a point may not be directly known precisely, but instead a region is directly available such that this point is guaranteed to lie within. Let $I = (S, U, A, w)$ be an instance of the maximal points under uncertainty problem. Then S is a set of points in the plane with their actual coordinates and U consists of the points that have uncertain location. For each $u \in U$ function w maps u to its actual x and y coordinates $w_u = (w_u^x, w_u^y)$ and function A maps u to a region in the plane. Here the set $\phi(S, U, w)$ actually only contains one element as, for the underlying problem, a set which contains all maximal points is unique. For the rest of this thesis concerning the maximal points under uncertainty problem we redefine an instance of the problem for convenience as $I = (P, \mathcal{A})$ where $P = S \cup \bigcup_{u \in \mathcal{U}} w_u$ and $\mathcal{A} = S \cup \bigcup_{u \in \mathcal{U}} A_u$. Assuming now that $P = \{p_1, \ldots, p_n\}$ and $\mathcal{A} = \{A_1, \ldots, A_n\}$ are ordered similarly, we note for every $1 \leq i \leq n$ that $p_i \in A_i$. Furthermore we do not refer to an update operation taking place on some $u \in U$ but rather on some $A_j \in \mathcal{A}$ which then changes the instance of the problem from $(P, \mathcal{A})$ to $(P, \mathcal{A}')$ such that $\mathcal{A}' = \{A_1, \ldots, A_{j-1}, \{p_j\}, A_{j+1}, \ldots, A_n\}$. But we should note that this does not apply for Chapter 7 as an update there does not necessarily retrieve the precise location of a point.

If an uncertainty area $A \in \mathcal{A}$ consists of a singleton set, we say A is *trivial* and otherwise A is *non-trivial*. The aim is to be able to produce the set of all points that are maximal among P based solely on the information of $\mathcal{A}$. If $\mathcal{A}$ is insufficient then at least one of its elements must be updated, which then reduces the updated uncertainty area to trivial. If then the resultant set is precise enough to calculate all maximal points, the recorded set which indicates which areas have been updated is 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.

In the adaptive online setting an algorithm is not given the set P initially, but only the set $\mathcal{A}$ and must perform updates on $\mathcal{A}$ continuously until an update solution has been achieved. In the non-adaptive online setting an algorithm is not able to modify $\mathcal{A}$ during the process, and thus an update solution has to be computed for the initial set $\mathcal{A}$. In the verification setting, however, an algorithm is also given a set of assumed precise location of points $P' = \{p'_1, \ldots, p'_n\}$ 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 every $1 \leq i \leq n$. As such, the aim in the verification setting is to produce an optimal update solution. Formally:

Definition 4.2. *A Maximal Point Verification problem, MPV for short, is a pair $(P, \mathcal{A})$, where P is a set of points and $\mathcal{A}$ is a set of uncertainty areas for P. The aim is to identify a minimal set of areas in $\mathcal{A}$, that when updated verifies the maximal points among P as maximal based on the information of $\mathcal{A}$ and the results of the updates.*

We extend the notion of a point being higher than another point to be also used on uncertainty areas:

Definition 4.3. *Let p be a point and A and B be uncertainty areas.*

- *If for every point $a \in A$ we have $p > a$ then $p > A$.*
- *If for every point $a \in A$ we have $a > p$ then $A > p$.*

- *If for every point $a \in A$ and every point $b \in B$ we have $a > b$ then $A > B$.*

We will further use the following notation:

Definition 4.4. *Let $\mathcal{A}$ be the set of all uncertainty areas with some $A \in \mathcal{A}$.*

- *We say A is maximal among $\mathcal{A}$ if for every point p that is higher than a point in A, p is not inside an area of $\mathcal{A} \setminus \{A\}$.*
- *We say A is dominated among $\mathcal{A}$ if for every point $p \in A$ there exists an area $B \in \mathcal{A} \setminus \{A\}$ such that $B > p$.*

We also note that an uncertainty area might be neither maximal nor dominated among a set of areas, whereas a point is either maximal or non-maximal among a set of points as given in Definition 4.1. If this is the case then the set of all maximal points cannot be computed. In other words:

Remark 4.5. *A maximal point under uncertainty problem is solved if and only if all areas in $\mathcal{A}$ are either maximal or dominated among $\mathcal{A}$.*

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

Chapter 5

Maximal Points Verification - Unrestricted Problem

5.1 Introduction

In this chapter we study the Maximal Point Verification problem (MPV), as given in Definition 4.2, where any type of uncertainty areas are allowed. We will use the maximal point notation as defined in Chapter 4. Our main result is, as stated later on in Theorem 5.8, that by a reduction from the Minimum Set Cover problem the MPV problem is also NP-hard. In our construction of the reduction each uncertainty area contains either a single point (i.e. the data is known precisely) or contains just two points. Hence, an MPV problem remains NP-hard even when restricted to areas of uncertainty that contain at most two points. It remains, however, open if the same holds when each uncertainty area is connected.

The effect of our result is significant for experimental evaluation of algorithms in the online and verification setting of the maximal point problem under uncertainty. It strengthens the role of constant competitive online algorithms, as they also represent a constant approximation algorithm for the verification setting. Finally it gives rise to find new restrictions on the uncertainty areas, such that the verification problem becomes solvable in polynomial time, and captures a large variety of applications for maximal points under uncertainty.

MPV example. In the example shown in Figure 5.1, the problem consists of three points p_1, p_2, p_3 and three areas A_1, A_2, A_3. Area A_2 consists of only point p_2 and hence A_2 is a trivial area and the location of p_2 is already verified. For every point in area A_1 there does not exist a point in A_2 or A_3 that is higher. Therefore, regardless of where p_1 lies in A_1 and where p_3 is located in A_3, point p_1 will be maximal among P. Based only on the areas of uncertainty, point p_3 may or may not be maximal among P. So updates have to be requested to verify some points, and therefore to make the problem solvable based on the initial areas of uncertainty and the information retrieved by the updates. The set $\{A_1, A_3\}$ is clearly an update solution as after updating these two sets the location of p_1 and p_3 are verified and both are maximal points in P. However the set $\{A_1\}$ is also an update solution as after verifying the location of p_1, neither p_1 nor p_2 are higher than any point in A_3. Hence even without verifying the location of p_3 within area A_3 both p_1 and p_3 must be both maximal among P. In this example the set $\{A_1\}$ is also an optimal update solution as without any update the maximal points cannot be calculated. We finish this example by noting that updating just A_3 is not an update solution. While this verifies the exact location of p_3, area A_1 still contains some points that are higher than p_3 and some that are not. So without also verifying the location of p_1 it is not clear whether p_3 is a maximal point among P or not.

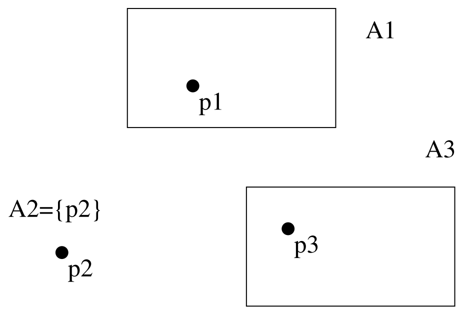


Figure 5.1: Example of an MPV problem

In the last part of this section we recall the classical Minimum Set Cover problem:

Definition 5.1. *An instance of the Minimum Set Cover (MSC) problem consists of the pair* $(U, \mathcal{S})$ *where* $U = \{1, \ldots, n\}$ *is the universe of* n *members and* $\mathcal{S} = \{S_1, \ldots, S_k\}$ *is a family of subsets of* U. *We say* $\mathcal{C} \subseteq \mathcal{S}$ *is a cover if* $\bigcup_{C \in \mathcal{C}} C = U$. *The aim is to compute a cover* $\mathcal{C}$ *such that* $|\mathcal{C}|$ *is minimal.*

5.2 Construction

In this section we give the construction of an MPV problem out of an MSC problem. We call the instance of the MSC problem MC $= (U, \mathcal{S})$ with $U = \{1, \ldots, n\}$ and $\mathcal{S} = \{S_1, \ldots, S_k\}$. The instance of the MPV will be denoted by MP.

The idea behind the construction is to have different types of areas in MP representing different aspects of MC. A set of areas (B's) will correspond to elements of U and another set of areas (A's) will correspond to elements of each $S_j \in \mathcal{S}$. The areas are positioned such that for each area corresponding to an element i of U, at least one area corresponding to the occurrence of i in the set S_j must be included in any update solution. With the help of another set of areas (D's), the areas corresponding to elements of a set S_j are linked together. So, if an update solution contains one area corresponding to an element of a set S_j the update solution can be modified to include all areas that correspond to elements of S_j without increasing the size of the update solution.

The construction is done by using three different types of gadgets, and each gadget is placed in its own rectangular region. These regions are located in the plane in such a way that no point in one gadget is higher than any point in another gadget. This can be achieved by placing all gadgets in regions diagonally top-left to bottom-right in the plane and we note the order of the placement is not important, see Figure 5.2.

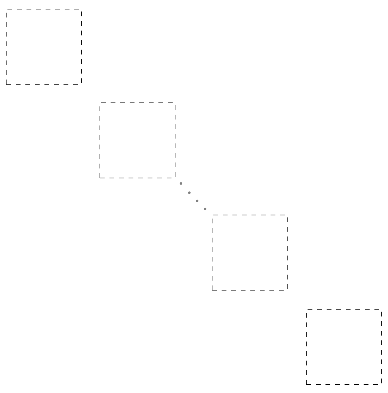


Figure 5.2: Placement of gadgets

Type 1 gadget. For each $i \in U$ there exists one gadget of type 1. This contains the point b_i, which is the lower left corner of the gadget, and multiple distinct points along the diagonal of the gadget. For each set $S_j \in \mathcal{S}$ that contains i, a point a_j^i is placed on the diagonal. See Figure 5.3.

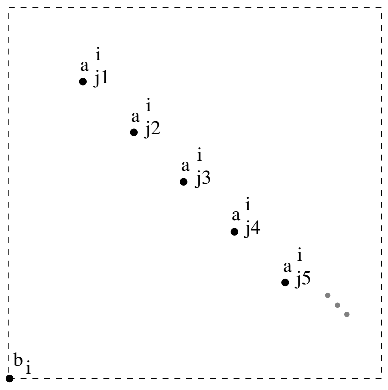


Figure 5.3: Type 1 gadget

Type 2 gadget. For each set $S_j \in S$ there exists one gadget of type 2. This contains for every $i \in S_j$ a point c_j^i along the diagonal of the gadget such that all points are pairwise distinct. In addition points $d_j^1, \ldots, d_j^t$ with $t = |S_j| - 1$ are placed in such a way that for each d_j^r with $1 \leq r \leq t$ there exist exactly two points c_j^i and $c_j^{i'}$ that are higher. Furthermore any two neighbouring points c_j^i and $c_j^{i'}$ are higher than exactly one point d_j^r. This can be done easily by placing the points $d_j^1, \ldots, d_j^t$ along a line that is parallel to the diagonal, and closer to the bottom-left corner of the gadget than the diagonal. See Figure 5.4.

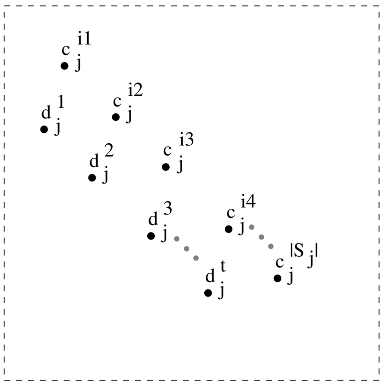


Figure 5.4: Type 2 gadget

Type 3 gadget. For each set $S_j \in \mathcal{S}$ there exists one gadget of type 3. This gadget just consists of $|S_j| - 1$ distinct points $e_j^1, \ldots, e_j^t$ placed along the diagonal. See Figure 5.5.

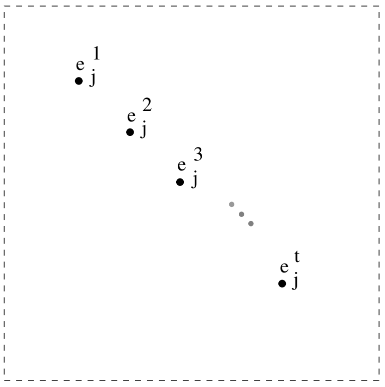


Figure 5.5: Type 3 gadget

The various points placed in the three gadgets, are now used to define the areas of uncertainty $\mathcal{A}$, and the set of precise location points P for MP.

Out of the points from the different gadgets we build the following sets where each set corresponds to an uncertainty area for MP. For every $i \in U$ let B_i be the set containing only b_i. For every $i \in U$ and $S_j \in \mathcal{S}$ with $i \in S_j$ let A_j^i be the set containing the two points a_j^i and c_j^i. For every $S_j \in \mathcal{S}$ and $1 \leq r \leq |S_j| - 1$ let D_j^r be the set containing the two points d_j^r and e_j^r.

To handle these sets better in the remaining part of this chapter we group some of these areas together. We say $A_j = \{A_j^i \mid i \in S_j\}$ and $D_j = \{D_j^1, \dots, D_j^t\}$ with $t = |S_j| - 1$. We also note that $|A_j| = |D_j| + 1 = |S_j|$.

Further we say B is the set of all areas that correspond to an element of U (or formally $B = \{B_1, \dots, B_n\}$), A is the set of all areas that correspond to an element of any set $S_j \in \mathcal{S}$ (or formally $A = \cup_{S_j \in \mathcal{S}} A_j$) and D is the set of all areas in any D_j (or formally $D = \cup_{S_j \in \mathcal{S}} D_j$).

This allows us to define our instance of the MPV in the following way: MP $= (P, \mathcal{A})$ with $\mathcal{A} = B \cup A \cup D$ and $P = \{b_1, \dots, b_n\} \cup \{a_j^i \mid i \in S_j\} \cup \{e_j^r \mid 1 \leq r \leq |S_j| - 1\}$.

We are now analysing the constructed problem MP and highlight properties that are needed in the further section.

Size of MP. There exist exactly n type 1 gadgets where each contains one point b_i with some $i \in U$. Each type 1 gadget contains at most k further points $\{a_j^i \mid i \in S_j\}$. There exist exactly k type 2 gadgets. Each contains at most $2n - 1$ points $c_j^1, \dots, c_j^{|S_j|}$ and $d_j^1, \dots, d_j^{|S_j|-1}$, since $|S_j| \leq n$. There exist exactly k type 3 gadgets. Each contains at most $n - 1$ points $e_j^1, \dots, e_j^{|S_j|-1}$ since $|S_j| \leq n$.

Hence for the MP constructed we have $n + 2k$ gadgets and at most $n * (1 + k) + k * (2n - 1) + k * (n - 1) = n + 4nk - 2k$ points. As each point only lies in one area of uncertainty also $|\mathcal{A}|$ is at most $n + 4nk - 2k$ and so the input size of MP is polynomial in the size of MC.

Maximal points among P. A point a_j^i for some j and i is part of a type 1 gadget and is clearly maximal among all points placed in the gadget. As two different gadgets are located so that no point of one is higher than a point of another, all points a_j^i are maximal among P. The same follows for the all points e_j^r in type 3 gadgets and therefore all such points are also maximal among P.

As for every $i \in U$ there must exists at least one $S_j \in \mathcal{S}$ with $i \in S_j$, by the construction of the type 1 gadget for i, also point a_j^i was added to that gadget. As all such points are higher than b_i, point b_i is non-maximal among P.

Maximal areas among $\mathcal{A}$. Each area in A consists of two points a_j^i and c_j^i. One is located inside a type 1 gadget and the other inside a type 2 gadget. For both points there is no area in $\mathcal{A}$ with a higher point, and therefore even without any updates all areas in A are maximal.

For each area $B_i \in B$ there exist areas in A which contain both a point higher and a point not higher than B_i. So area B_i is neither maximal nor dominated among $\mathcal{A}$ and, by Remark 4.5, further updates are needed.

Each area in D has two points. One is located in a type 3 gadget for which there is no point from some other area higher. The other is located in a type 2 gadget where there are two areas in A that contain points that are higher. So among $\mathcal{A}$ it is neither maximal nor dominated and further updates are needed.

Update solutions for MP. Following the above analysis of maximal among $\mathcal{A}$ areas, we give the following remark:

Remark 5.2. *A set of areas is an update solution if and only if it contains for each i an area $A^i_j \in A$ for some j, and also for each area in D either this area or the two areas in A that are potentially higher.*

Following from this only updates of areas in A_j and D_j will help to identify areas of D_j as maximal among $\mathcal{A}$. Based on the construction of type 2 gadgets, updating k areas of A_j can at most identify $k-1$ areas of D_j as maximal among $\mathcal{A}$. Hence the smallest update set that identifies all areas of D_j as maximal among $\mathcal{A}$, is D_j itself. Any other set of updates must be bigger. Formally:

Remark 5.3. *Let R be an update solution. Then for every j the set R must contain either D_j or it must contain at least $|D_j|+1$ areas of $D_j \cup A_j$.*

This leads to the following Lemma:

Lemma 5.4. *Let R be an update solution for MP and let $A^i_j \in R$ for some j and i be an area. Then $R' = (R \setminus D_j) \cup A_j$ is also an update solution with $|R'| \leq |R|$.*

Proof. Since R is an update solution, by Remark 5.2 for every $i \in U$ the set R must contain an area $A^i_{j'}$ for some j'. As R' is constructed by potentially removing areas of D and adding areas of A the set R' must also contain area $A^i_{j'}$.

Let $D^r_{j''} \in D$. Again by Remark 5.2 either $D^r_{j''} \in R$ or the two areas in A that are potentially higher than $D^r_{j''}$ are in R. If R contains the two areas in A that are potentially higher than $D^r_{j''}$ then also R' must contain these areas as no area in A was removed when creating R'. If $D^r_{j''} \in R$ also R' must contain $D^r_{j''}$ unless $j'' = j$. In that case as all areas in A_j were added to R', it must also contain the two areas in A_j that are potentially higher than $D^r_{j''}$. Hence by Remark 5.2 also R' is an update solution.

We now show that $|R'| \leq |R|$. By our assumption $A^i_j \in R$. So as R already contains at least one element of A_j, there will be at most $|A_j|-1$ elements of A_j added to R'. We noted in the construction of MP that $|A_j| = |D_j|+1$. Therefore it must follow that $|(R \setminus D_j) \cup A_j| \leq |R|$. □

5.3 Relating Update Solutions to Covers

In this section we show how to construct a cover of MC out of an update solution of MP and vice versa. We will also note how the size of the update solutions and covers relate to each other.

From update solution to cover. Let R be an update solution for MP.

Before creating the cover we create a different update solution R'. The set R' is based on R but for every j such that there exists an i with $A_j^i \in R$ all potential areas of D_j are removed from R and all areas of A_j are added. By Lemma 5.4, we have that R' is also an update solution with no greater size than R. Furthermore by doing so, the update solution R' contains for every index j either the set A_j or D_j but never a mixture.

The cover $\mathcal{C}$ is constructed based on R' in the following way. For each index j such that $A_j \subseteq R'$ we choose the set $S_j \in \mathcal{S}$ to be included in $\mathcal{C}$ and otherwise not.

This is denoted as:

$$\mathcal{C} = \{S_j \in \mathcal{S} \mid A_j \subseteq R'\}$$

We now show that $\mathcal{C}$ is a cover, in other words that every element of U is found in at least one set of $\mathcal{C}$.

Let some $i \in U$ for the MC. Then in MP there exists area B_i. By Remark 5.2 there exists a j with $A_j^i \in R'$. Since this area A_j^i was constructed in the creation of MP we have that $i \in S_j$. As A_j^i is also in R' the set A_j must be a subset of R and $S_j \in \mathcal{C}$.

We note that the construction of $\mathcal{C}$ is done in polynomial time and the sizes of R, R' and C relate to each other in the following way.

By the construction of R' we have:

$$|R'| = \sum_{A_j \subseteq R'} |A_j| \ + \sum_{A_j \not\subseteq R'} |D_j|$$

As $|A_j| = |D_j| + 1 = |S_j|$ for every j we get by the construction of $\mathcal{C}$ that:

$$|R'| = \sum_{S_j \in \mathcal{C}} |S_j| \ + \sum_{S_j \in \mathcal{S} \setminus \mathcal{C}} (|S_j| - 1)$$

$$= |\mathcal{C}| + \sum_{S_j \in \mathcal{C}} (|S_j| - 1) \ + \sum_{S_j \in \mathcal{S} \setminus \mathcal{C}} (|S_j| - 1)$$

$$= |\mathcal{C}| + \sum_{S_j \in \mathcal{S}} (|S_j| - 1)$$

Since by Lemma 5.4 we get $|R| \geq |R'|$ we have:

$$|R| \geq |\mathcal{C}| \ + \sum_{S_j \in \mathcal{S}} (|S_j| - 1)$$

We summarise our results on the construction of $\mathcal{C}$ in the following Lemma:

Lemma 5.5. *Let R be an update solution for MP. Then a cover of MC can be* *constructed in polynomial time with* $|R| \geq |\mathcal{C}| + \sum_{S_j \in \mathcal{S}} (|S_j| - 1)$.

From cover to update solution. Similarly to the construction of a cover for MC out of a given update solution of MP, we now show how to construct an update solution for MP out of a given cover for MC.

Let $\mathcal{C}$ be a cover for MC.

The set R of areas in MP is based on $\mathcal{C}$ as follows: For each index j such that $S_j \in \mathcal{C}$ we choose the set A_j to be included in R. For each index j such that $S_j \in (\mathcal{S} \setminus \mathcal{C})$ we choose the set D_j to be included in R.

This is denoted as:

$$R = (\bigcup_{S_j \in \mathcal{C}} A_j) \cup (\bigcup_{S_j \in (\mathcal{S} \setminus \mathcal{C})} D_j)$$

We note the following: Firstly, let $i \in U$. Since $\mathcal{C}$ is a cover there exists a j such that $S_j \in \mathcal{C}$ and $i \in S_j$. Hence, by the construction of MP, area A^i_j exists in MP. As $S_j \in \mathcal{C}$ we have that $A_j \subseteq R$ and in particular $A^i_j \in R$.

Secondly, let $D^r_j \in D$. If $S_j \notin \mathcal{C}$ the set R contains D_j and therefore also D^r_j. Otherwise R contains A_j and therefore also the two areas in A_j that are potentially higher than D^r_j.

So R satisfies both condition of Remark 5.2 and is hence an update solution for MP.

We recall from the MP-construction that for every set S_j there is a set A_j and a set D_j such that $|A_j| = |D_j| + 1 = |S_j|$. So,

$$\begin{aligned}
|R| &= \sum_{S_j \in \mathcal{C}} |S_j| \ + \sum_{S_j \in \mathcal{S} \setminus \mathcal{C}} (|S_j| - 1) \\
&= |\mathcal{C}| + \sum_{S_j \in \mathcal{C}} (|S_j| - 1) \ + \sum_{S_j \in \mathcal{S} \setminus \mathcal{C}} (|S_j| - 1) \\
&= |\mathcal{C}| + \sum_{S_j \in \mathcal{S}} (|S_j| - 1)
\end{aligned}$$

This leads to the following lemma:

Lemma 5.6. *Let $\mathcal{C}$ be a cover of MC. Then there exists an update solution R for MP with $|R| = |\mathcal{C}| + \sum_{S_j \in \mathcal{S}} (|S_j| - 1)$.*

5.4 NP-Hardness Proof

We have shown so far how an instance MP of the Maximal Point Verification problem can be constructed out of an instance MC of the Minimum Set Cover problem, how one can build a solution for one of these two problem instances based on the solution of the other, and how the sizes of the solutions are related. We now argue that an optimal update solution corresponds to a minimal cover.

Lemma 5.7. *Let R be an optimal update solution for MP. Then the cover $\mathcal{C}$ constructed out of R is a minimal cover for MC.*

Proof. Let's assume there exists a cover $\overline{\mathcal{C}}$ for MC such that $|\overline{\mathcal{C}}| < |\mathcal{C}|$. Further let $\overline{R}$ be the update solution for MP constructed from $\overline{\mathcal{C}}$ as shown in Section 5.3.

By Lemma 5.5:

$$|\mathcal{C}| \leq |R| - \sum_{S_j \in \mathcal{S}} (|S_j| - 1)$$

And by Lemma 5.6:

$$|\overline{\mathcal{C}}| = |\overline{R}| - \sum_{S_j \in \mathcal{S}} (|S_j| - 1).$$

Since $|\overline{\mathcal{C}}| < |\mathcal{C}|$ so must $|\overline{R}| < |R|$. This is a contradiction as R was an optimal update solution. So $\mathcal{C}$ must be a minimal cover of MC. □

We are using the established results to prove Theorem 5.8.

Theorem 5.8. *Solving the Maximal Point Verification problem is NP-hard.*

Proof. In Section 5.2 we have presented the construction of a MPV problem for a given MSC problem. As noted in Section 5.2 the size of the MPV problem is polynomial in the size of the MSC problem and also choosing coordinate values for the points is fairly trivial. So the construction can be done in polynomial time.

By Lemma 5.7 a solution of the MPV can be used to construct a solution of the MSC problem. As remarked in Section 5.3 that construction is polynomial in the size of the MPV problem.

Hence, if the MPV problem is solvable in polynomial time, then this must also be the case for the MSC problem. The MSC problem is shown to be NP-hard in [34] and therefore the MPV problem is also NP-hard. □

Chapter 6

Maximal Points - Direct Product

6.1 Introduction

In this chapter we consider the maximal point under uncertainty problem with a new restriction. We will use again the maximal point notation as defined in Chapter 4. As in Chapter 4 and Chapter 5, we are still working with 2-dimensional points. However, in this chapter, instead of restricting to either trivial areas or connected, open areas, we restrict to areas that are the direct product of uncertain areas in the x and y coordinates. Note that this is incomparable with the restriction of Bruce et al. [6]. On the one hand, this restriction allows disconnected areas of uncertainty. On the other hand, the shapes of uncertainty areas are restricted to be a specific kind of union of rectangles.

Our contribution is the new restriction to the problem which is update competitive for the online case. We also show that there exists a polynomial time algorithm that solves the verification problem. In this section we give the motivation and the definition for the new restriction. In Section 6.2 we give an example of the problem. In Section 6.3 we give details about the process of classifying precise locations among P and uncertainty areas among $\mathcal{A}$. In Section 6.4 we show that the witness algorithm by Bruce et al. [6], for the online adaptive setting of the problem, is also 3-update competitive for our new restriction. And finally in Section 6.5 we analyse our algorithm for the verification problem under the new restriction. The effect of our results for the verification setting is significant for experimental evaluation of algorithms in the online adaptive setting of the maximal point problem under uncertainty.

Background - Motivation. The problem of finding maximal points under uncertainty was introduced by Bruce et al. [6] and an update strategy with optimal competitive ratio 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 in [6] is to have only uncertainty areas which are either trivial or closures of connected, open areas. Further study of the problem by us in Chapter 5 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. Following the previous results for maximal points under uncertainty, 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 uncertainty areas which the restriction by Bruce et al. was unable to do so.

Although the restricted problem as studied by Bruce et al. yields nice results, it does not capture scenarios where the uncertain information about an area may contain non-consecutive values. We want to have a model where the categorized information does not have to be continuous. Furthermore, while uncertain information may contain non-consecutive values, often this information is categorized into properties which are independent of each other.

This scenario can be seen where, given a list of products or services, we want to distinguish all the options that are as good or better from all the others. Each product or service has two properties, which are of the same type as all others considered, and a rough estimate is effortlessly available for which the precise measurement can be obtained but for some reason doing this involves some sort of cost/effort. A product or service is distinguished if, with the current information, we can establish that no other is better in one property and at least as good in the other.

Real life example 1. Given a list of similar products, where each has a price and rating, one wants to distinguish those that are maximal in terms of low price and high rating. A product may have an uncertain price in non-consecutive ranges depending on availability from different sellers, time of purchase, total number or combination of products purchased, etc. An update retrieves the precise price and rating by, for example, visiting a store that can provide it. Independence of properties can be observed here as when comparing products the price of a product should not affect its rating.

Real life example 2. Given a list of hotels with distance to the beach (the closer the more desirable) and rating of service (the higher the more desirable), one wants to distinguish those that are maximal in terms of close proximity to the beach and rating of service. A hotel may have an uncertain rating given in non-consecutive ranges depending on when a survey was last taken, or it might fluctuate significantly between the high and low seasons. Furthermore the distance from a hotel to the beach might be uncertain due to GPS accuracy factors, or even the specific location not at all available at times as the coastline might be closed to the public due to coastal hazards (e.g pollution, trawling, weather condition). An update retrieves the precise distance to the beach by, for example, verifying it from google maps and checking the coastal condition for the specific dates considered. The update further retrieves the rating of service by, for example, retrieving data from multiple sources instead of just one and then producing an average, or even narrow the results to consider only recorded data for similar dates in previous years. Independence of properties can be observed here as when comparing hotels the distance of a hotel to the beach should not affect its service rating.

Real life example 3. Two people wanting to stay together are searching for a place. These people have to commute to their individual places of work. They want to find a place to live that is as close as possible to both places of work. From their options they would like to eliminate those for which there exists some other option that is at least as good for both of them. For each location they can get a rough estimate on commuting time. An update can be the actual test run which would give the precise value. Independence of properties can be observed here as of course a place for which the commuting time is low for one person is not necessarily low for the other.

With the maximal points under uncertainty in mind, independence between the x and y coordinates can be modelled as constructing an area of uncertainty from the direct product of two sets; one containing x coordinate values and another containing y coordinate values. This further allows non-consecutive values. This kind of restriction captures problems like the three real life examples mentioned. 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.

Considering the above we introduce the following restriction for the accepted input areas for the maximal points under uncertainty problem. Given that we are still studying the problem in 2-dimensional 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 6.1. *An instance of the Maximal Point Direct Product (MPDP) problem consists of the pair $(P, \mathcal{A})$ such that:*

- $P = \{p_1, p_2, \ldots, p_n\}$ *is the set of all precise location points*
- $\mathcal{A} = \{A_1, A_2, \ldots, A_n\}$ *is the set of all uncertainty areas where $p_i \in A_i$ for every $1 \leq i \leq n$ such that each area A_i is the direct product $\{A_i^x \times A_i^y\}$ where A_i^x is a set of x values and A_i^y is a set of y values.*

Given for example uncertain values for x-coordinate $A_i^x = \{1, [2, 4]\}$ and y-coordinate $A_i^y = \{2, [4, 6]\}$, area A_i can be depicted as shown in Figure 6.1. Note that the upper and lower limits of both A_i^x and A_i^y are closed in this example but this is not a restriction throughout this chapter.

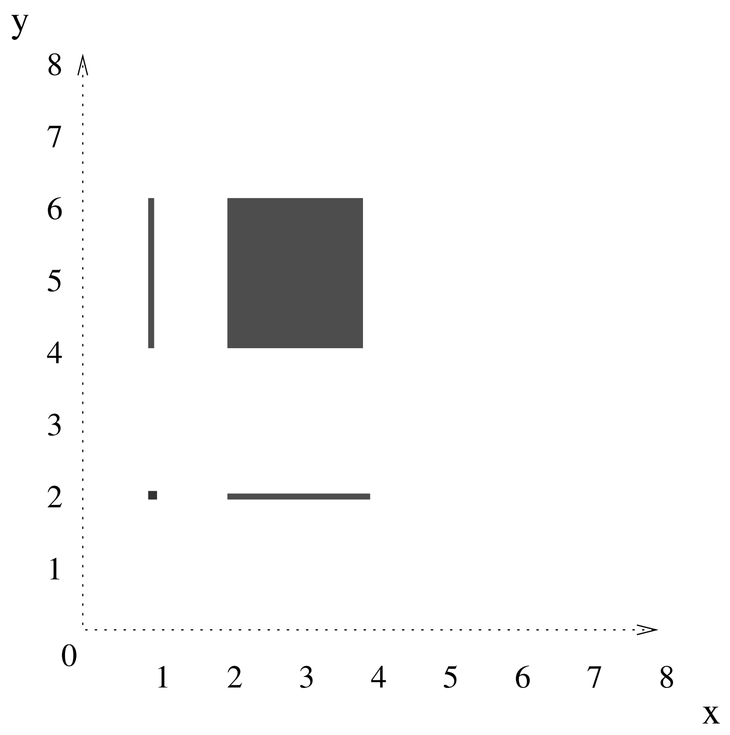


Figure 6.1: Example of area $A_i = \{1, [2, 4]\} \times \{2, [4, 6]\}$

6.2 MPDP Example

Figure 6.2 illustrates an instance $(P, \mathcal{A})$ of the MPDP problem where it consists of the three points p_1, p_2, p_3 and the three uncertainty areas A_1, A_2, A_3 such that $\mathcal{A} = \{A_1, A_2, A_3\}$ and $P = \{p_1, p_2, p_3\}$ and by definition $p_1 \in A_1$, $p_2 \in A_2$, $p_3 \in A_3$. As shown area A_3 consists of two disconnected regions. Furthermore area A_2 overlaps with area A_1 and area A_3 overlaps with area A_1, or in other words they share some points about where their precise location might be. We note that updating an area which overlaps with some other area does not reduce the uncertain information of the latter, for example updating area A_3 does not affect the uncertain information of A_1 about where p_1 might be located.

We can see that points p_2 and p_3 are both maximal among P since there is no point in P that is higher than either of them. Also p_1 is non-maximal among P since p_2 and p_3 are higher. However without verifying any of the points in P (i.e. updating the corresponding area which in turn reduces its size to trivial), 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 area of uncertainty A_1 and the points p_2 and p_3 (Figure 6.3). Updating only A_1 on the other hand validates that p_1 is indeed non-maximal, since A_1 is now reduced to trivial size and wherever p_2 lies in A_2, point p_2 will always be higher. This brings us to Figure 6.4 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 point higher than a point in A_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 update solutions $\{A_1, A_2\}$ and $\{A_1, A_3\}$. These update solutions area also optimal because by our previous reasoning area A_1 must be updated and also updating only A_1 does not suffice.

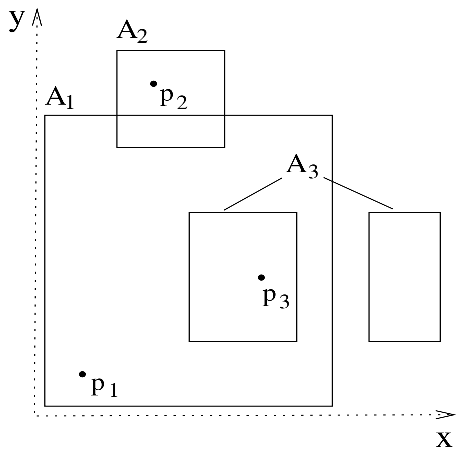


Figure 6.2: Example of an MPDP problem

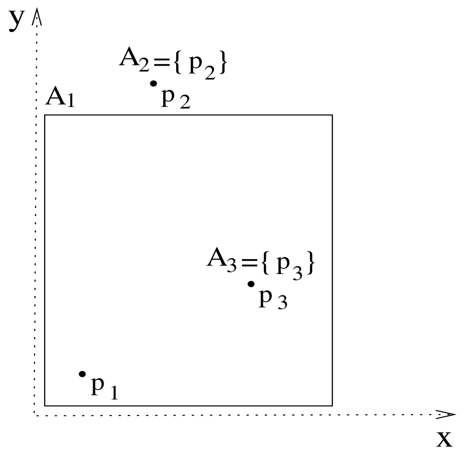


Figure 6.3: Updating A2, A3

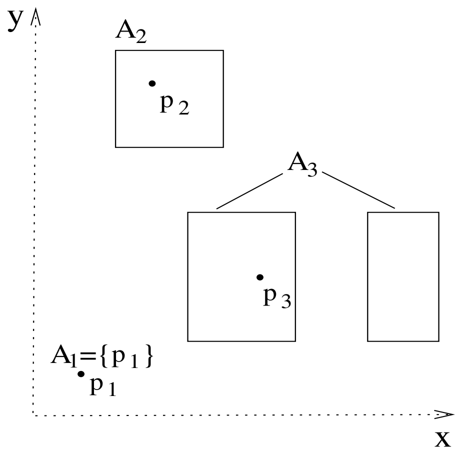


Figure 6.4: Updating A1

6.3 Classifying Precise Locations and Uncertainty Areas

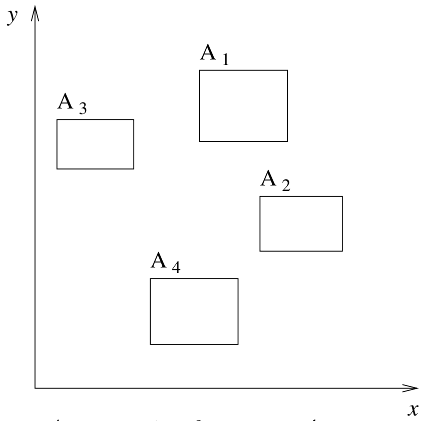
In this section we give two more classifications of uncertainty areas which we will use in Chapter 6, 7 and 8. Furthermore we give the process for determining the status of a point among P and the status of an uncertainty area among $\mathcal{A}$. Finally we note the time complexity for each of the two processes.

We have a closer look at the uncertainty areas which are neither maximal nor dominated among $\mathcal{A}$. These areas can be further divided into the following two classifications:

Definition 6.2. *Let $\mathcal{A}$ be the set of all uncertainty areas with some $A \in \mathcal{A}$. If A is neither maximal nor dominated among $\mathcal{A}$:*

- *We say A is partly among $\mathcal{A}$, if further there exists a point $a \in A$ such that there does not exist a point in any areas in $\mathcal{A} \setminus \{A\}$ that is higher than a.*
- *We say A is dependent among $\mathcal{A}$, if further for every point $a \in A$ there exists an area $B \in \mathcal{A} \setminus \{A\}$ with a point $b \in B$ such that $b > a$.*

In Figure 6.5 we can see an example with the four different classifications of uncertainty areas $\mathcal{A} = \{A_1, A_2, A_3, A_4\}$. Note that the areas here appear rectangular and connected just for ease and this is not a restriction throughout Chapter 6, 7 and 8 unless specifically stated.



A_1 : maximal among $\mathcal{A}$
A_2 : partly among $\mathcal{A}$
A_3 : dependent among $\mathcal{A}$
A_4 : dominated among $\mathcal{A}$

Figure 6.5: Classification of uncertainty areas $\mathcal{A} = \{A_1, A_2, A_3, A_4\}$

6.3.1 Status of Points Among P

To determine if the status of a precise location $p \in P$ is either maximal or non-maximal among P, we have to compare the x and y coordinates of p against at most all other points in $P \setminus \{p\}$. Specifically:

- Point $p = (p_x, p_y)$ is maximal among P if every $q = (q_x, q_y)$ in P is such that either $p_x > q_x$ or $p_y > q_y$ or $p = q$.
- Point $p = (p_x, p_y)$ is non-maximal among P if there exists $q = (q_x, q_y)$ in P such that $q_x \geq p_x$ and $q_y \geq p_y$ and $q \neq p$.

We also note:

Remark 6.3. *Determining the status of all points in P is done in polynomial time.*

6.3.2 Status of Uncertainty Areas Among $\mathcal{A}$

Let A_1 be an uncertainty area of a given instance $(P, \mathcal{A})$ of the MPDP problem. By Definition 6.1, area A_1 is made up by the direct product of a coordinate x values set A_1^x and a coordinate y values set A_1^y. Therefore A_1^x and A_1^y are available out of which we can obtain their upper and lower limits, as well as whether each limit is inside the set or not (i.e. open or closed limits). We denote the lower and upper limits of A_1^x as l_1^x and h_1^x respectively. Similarly we denote the lower and upper limits of A_1^y as l_1^y and h_1^y respectively. We note that if A_1 is trivial then all limits are closed and $l_1^x = h_1^x$ and $l_1^y = h_1^y$. These four values for each uncertainty area, along with whether they are inside their respective sets, is essentially all the information needed to know in order to determine the status of an area among $\mathcal{A}$.

Specifically we give the cases for each of the four classifications for an uncertainty area $A_i \in \mathcal{A}$:

- A_i is maximal among $\mathcal{A}$ if for every $A_j \in \mathcal{A} \setminus \{A_i\}$:
  - $l_i^x > h_j^x$ or $l_i^y > h_j^y$ or
  - $l_i^x = h_j^x$ and either l_i^x or h_j^x is open or
  - $l_i^y = h_j^y$ and either l_i^y or h_j^y is open or
  - $l_i^x = h_j^x$ and $l_i^y = h_j^y$ and all of l_i^x, l_i^y, h_j^x, h_j^y are closed
- A_i is dominated among $\mathcal{A}$ if there exists $A_j \in \mathcal{A} \setminus \{A_i\}$ such that:
  - $l_j^x \geq h_i^x$ and $l_j^y > h_i^y$ or
  - $l_j^x > h_i^x$ and $l_j^y \geq h_i^y$ or

  - $l_j^x = h_i^x$ and $l_j^y = h_i^y$ and one of l_j^x, l_j^y, h_i^x, h_i^y is open

- A_i is partly among $\mathcal{A}$ if:
  - for every $A_j \in \mathcal{A} \setminus \{A_i\}$:
    - $h_i^x > h_j^x$ or $h_i^y > h_j^y$ or
    - $h_i^x = h_j^x$ and $h_i^y = h_j^y$ and
      - all of h_i^x, h_i^y, h_j^x, h_j^y are open or
      - both h_i^x and h_i^y are closed or
      - h_i^x is closed and h_j^x is open or
      - h_i^y is closed and h_j^y is open or
      - h_i^x is closed, h_i^y is open, h_j^x is closed and h_j^y is open or
      - h_i^x is open, h_i^y is closed, h_j^x is open and h_j^y is closed
  - and furthermore there exists $A_j \in \mathcal{A} \setminus \{A_i\}$ such that:
    - $h_j^x > l_i^x$ and $h_j^y > l_i^y$ or
    - $l_i^x = h_j^x$ and both l_i^x, h_j^x are closed and $h_j^y > l_i^y$ or
    - $l_i^y = h_j^y$ and both l_i^y, h_j^y are closed and $h_j^x > l_i^x$

- A_i is dependent among $\mathcal{A}$ if:
  - for every $A_j \in \mathcal{A} \setminus \{A_i\}$:
    - $h_i^x > l_j^x$ or $h_i^y > l_j^y$ or
    - $h_i^x = l_j^x$ and $h_i^y = l_j^y$ and all of h_i^x, h_i^y, l_j^x, l_j^y are closed
  - and furthermore there exists $A_j \in \mathcal{A} \setminus \{A_i\}$ such that:
    - $h_j^x > h_i^x$ and $h_j^y > h_i^y$ or
    - $h_j^x = h_i^x$ and $h_j^y = h_i^y$ and both h_j^x, h_j^y are closed and either h_i^x or h_i^y is open or
    - $h_j^x = h_i^x$ and $h_j^y > h_i^y$ and h_j^x is closed or
    - $h_j^x = h_i^x$ and $h_j^y > h_i^y$ and both h_j^x, h_i^x are open or
    - $h_j^y = h_i^y$ and $h_j^x > h_i^x$ and h_j^y is closed or
    - $h_j^y = h_i^y$ and $h_j^y > h_i^y$ and both h_j^y, h_i^y are open

We also note:

Remark 6.4. *Determining the status of all uncertainty areas in $\mathcal{A}$ is done in polynomial time.*

6.4 Online Problem

The restriction of the maximal point problem as studied by Bruce et al. [6] is that the areas of uncertainty are either trivial or closures of connected open areas. It is interesting that our restriction allows disconnected areas, whereas this is not the case in the restriction proposed in [6]. In Subsection 6.4.1 we show that the 3-update competitive witness algorithm of Bruce et al. also applies for the online setting of the MPDP problem. Furthermore it is shown that this is the best possible by any deterministic online algorithm in Subsection 6.4.2, which then brings us to Theorem 6.12 in Subsection 6.4.3.

6.4.1 Upper Bound

Overview. In this subsection we use the material as in [6], that is the witness algorithm for the maximal points under uncertainty problem (given in Algorithm 2) and a series of lemmas used to establish its update competitiveness. The proofs for the lemmas, however, are our own adapted for the MPDP problem. Lemma 6.6 and 6.9 are used to establish witness sets for the two cases of Algorithm 2 (recall from Chapter 2 a witness set is a set of updates such that every update solution must contain at least one of those). Furthermore Lemma 6.7 and Lemma 6.8 are used to help the proof of Lemma 6.9. Moreover we give Definition 6.5 for when a horizontal or vertical line splits an area which is being used for Lemma 6.8 and consequently Lemma 6.9.

We begin with stating the algorithm:

Algorithm 2 Witness algorithm for Maximal Points under Uncertainty

```
1: while There exists at least one partly or one dependent area among $\mathcal{A}$ do
2:     if There exists a partly area among $\mathcal{A}$ then
3:         Find a witness set $W$ ; Update all areas in $W$
4:     else(There must exists a dependent area among $\mathcal{A}$)
5:         Find a witness set $W$ ; Update all areas in $W$
6:     end if
7: end while
```

The split in these two cases helps to identify witness sets. Note that the idea is to concentrate first on areas that are partly among $\mathcal{A}$ and witness sets concerning these areas. Only if there are such areas left in the given instance, will a strategy based on the existence of dependent among $\mathcal{A}$ areas be used to find witness sets.

We now give our own definition for when a horizontal or vertical line splits an uncertainty area, which will be used later on for the proof of Lemma 6.9:

Definition 6.5. *Let l_1 be a vertical line with equation $x = n$ and let l_2 be a horizontal line with equation $y = m$. Further let A be an uncertainty area. If there exist two points $a = (a_x, a_y)$ and $b = (b_x, b_y)$ in A such that $a_x \geq n > b_x$ then point b exist also with $a_y = b_y$ (due to the direct product restriction as in Definition 6.1) and we say l_1 splits A. Similarly if $a_y \geq m > b_y$ then also $a_x = b_x$ and we say l_2 splits A.*

The following three lemmas (Lemma 6.6 , Lemma 6.8 and Lemma 6.9) are taken from [6] and adapted for the MPDP problem. Lemma 3 in [6] states that if there exists an area that is partly among $\mathcal{A}$ then there exists a witness set of size at most 2. This is equivalent to our Lemma 6.6:

Lemma 6.6. *Let A be a partly among $\mathcal{A}$ area, then there exists a witness set of size at most* 2*.*

Proof. Since A is partly among $\mathcal{A}$, it must contain a point h such that there does not exist a point in any area of $\mathcal{A} \setminus \{A\}$ that is higher than h (as otherwise it would be either dependent or non-maximal instead of partly among $\mathcal{A}$). For the same reason it must further contain another point l such that there exists at least one point b in an area $B \in \mathcal{A} \setminus \{A\}$ with $b > l$ (as otherwise it would be maximal instead of partly among $\mathcal{A}$). We note that it is possible that every point in B is higher than l, and not just point b. In other words potentially $B > l$. We now separate the two cases where either $B > l$ (Case 1) or $B \not> l$ (Case 2).

Case 1: $B > l$. By not updating area A, both points h and l remain in A. As $B > l$ then even after updating area B, the condition $B > l$ remains. We also know that there is no point in any area of $\mathcal{A} \setminus \{A\}$ that is higher than point h. Therefore without updating A itself this area cannot change its status to either maximal or dominated among $\mathcal{A}$, as needed for an update solution by Remark 4.5. Hence $\{A\}$ is a witness set. See Figure 6.6.

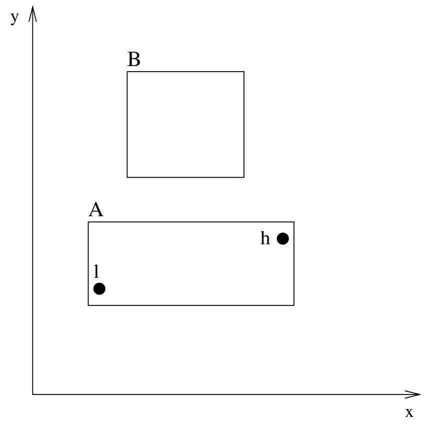


Figure 6.6: Case $B > l$

Case 2: $B \not> l$ (but still $b > l$). With a similar reasoning as for Case 1, by only updating areas in $\mathcal{A} \setminus \{A, B\}$ the points $b \in B$ and $l \in A$ remain such that $b > l$.

Therefore, as point $h \in A$ also remains, at least A or B has to be updated to change the status of area A to either maximal or dominated among $\mathcal{A}$. Hence $\{A, B\}$ is a witness set. See Figure 6.7.

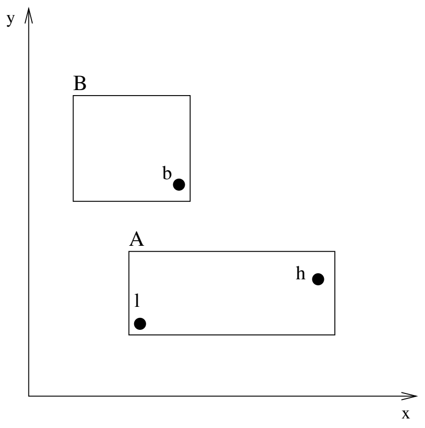


Figure 6.7: Case $b > l$

□

We note a general property for dominated among $\mathcal{A}$ areas which we will use later on:

Lemma 6.7. *Let area A be dominated among $\mathcal{A}$ and let B be another area. If there exists a point $a \in A$ such that $a > B$ then area B is also dominated among $\mathcal{A}$.*

Proof. Since A is dominated among $\mathcal{A}$ then there exists an area $C \in \mathcal{A} \setminus \{A\}$ with $C > A$ and as $a \in A$ then also $C > a$. As $C > a$ and $a > B$ then it must follow $C > B$. Hence B is also dominated among $\mathcal{A}$. □

Lemma 4 and Lemma 5 in [6] state that if there are no partly among $\mathcal{A}$ areas but there exists a dependent among $\mathcal{A}$ area, then there exists a witness set of size at most 3. This is equivalent to our Lemma 6.8 and Lemma 6.9 respectively:

Lemma 6.8. *A horizontal or vertical line can split at most one area that is maximal among $\mathcal{A}$.*

Proof. Let l_1 be a vertical line with equation $x = n$ and let A_1 and A_2 be areas maximal among $\mathcal{A}$. Further let l_1 split both areas A_1 and A_2.

As l_1 splits area A_1 then there exist points $a_1 = (a_1^x, a_1^y)$ and $b_1 = (b_1^x, b_1^y)$ such that $a_1, b_1 \in A_1$ where $a_1^x \geq n > b_1^x$ and $a_1^y = b_1^y$. It also follows that $a_1^x > b_1^x$.

As l_1 also splits area A_2, with the similar reasoning as for A_1, there exist points $a_2 = (a_2^x, a_2^y)$ and $b_2 = (b_2^x, b_2^y)$ such that $a_2, b_2 \in A_2$ where $a_2^x \geq n > b_2^x$ and $a_2^y = b_2^y$. It also follows $a_2^x > b_2^x$.

We now have that either $a_1^y \geq a_2^y = b_2^y$ or $a_2^y \geq a_1^y = b_1^y$, which follows that either $a_1^y \geq b_2^y$ or $a_2^y \geq b_1^y$.

Furthermore as $a_1^x \geq n > b_1^x$ and $a_2^x \geq n > b_2^x$ then also $a_2^x > b_1^x$ and $a_1^x > b_2^x$.

With the above properties we have that either $a_1^x > b_2^x \ \wedge \ a_1^y \geq b_2^y$ or $a_2^x > b_1^x \ \wedge \ a_2^y \geq b_1^y$ must be true. Therefore either $a_1 > b_2$ or $a_2 > b_1$. This is a contradiction as both areas A_1 and A_2 are maximal among $\mathcal{A}$ and therefore there must be no point in areas $\mathcal{A} \setminus \{A_1\}$ higher than b_1, and no point in areas $\mathcal{A} \setminus \{A_2\}$ higher than b_2. As such if l splits A_1 and A_2, one of them cannot be maximal among $\mathcal{A}$.

In a similar way if l_2 is a horizontal line with equation $y = m$ and splits two areas A_1 and A_2, then there exist points $a_1, b_1 \in A_1$ and $a_2, b_2 \in A_2$ such that $a_1^x \geq a_2^x = b_2^x$ or $a_2^x \geq a_1^x = b_1^x$ holds. As a_1, a_2 would lie on or above of l_2 and b_1, b_2 below l_2, then it follows that either $a_2 > b_1$ or $a_1 > b_2$. Therefore A_1 and A_2 cannot be both maximal among $\mathcal{A}$.

□

Lemma 6.9. *If there are no partly areas but there exists a dependent area among $\mathcal{A}$, then there exists a witness set of size at most* 3.

Proof. Assume there are no partly areas among $\mathcal{A}$ and 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 $h_i = (h_i^x, h_i^y)$ be the point made by the upper limits of the coordinate sets that make the direct product for A_i. We note that either $h_i \in A_i$ (upper limits for both x and y are closed) or $h_i \notin A_i$ (upper limit for either x or y is open). 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 these lines. 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 Q contains all the points that are higher than A_i. Also let $\overline{Q}$ denote the complement of Q. We show later in Figure 6.8 examples. The bounds of Q (lines l_1 and l_2) are shown with dotted lines, and non-inclusive limits of an uncertainty area are shown with dashed lines.

Since A_i is dependent among $\mathcal{A}$, there exists an area $A_j \in \mathcal{A} \setminus \{A_i\}$ with a point $a_j \in A_j$ such that $a_j > A_i$. If this was not the case, then A_i would be partly among $\mathcal{A}$ as it would contain a point for which there does not exist a point in some other area such that is higher. As all points that are higher than A_i are inside Q, then $a_j \in Q$ and hence $A_j \cap Q \neq \emptyset$.

Furthermore there must exist another point $b_j \in A_j$ such that $b_j \not> A_i$, as otherwise $A_j > A_i$ and therefore A_i would be dominated instead of dependent among $\mathcal{A}$. Hence also $A_j \cap \overline{Q} \neq \emptyset$.

We show examples of A_i and A_j in Figure 6.8:

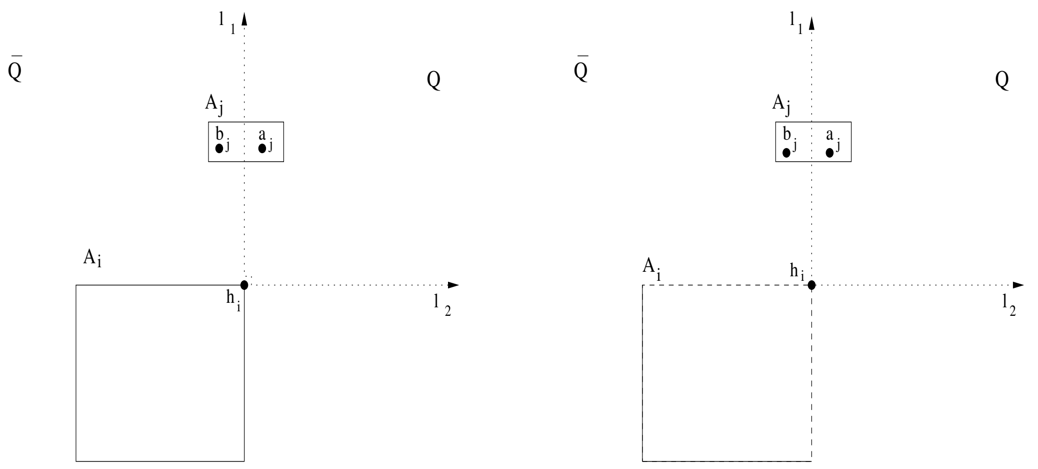


Figure 6.8: Construction of Q with either $h_i \in A_i$ or $h_i \notin A_i$

By our assumptions there are no other dependent among $\mathcal{A}$ areas with a point higher than A_i, and also no partly among $\mathcal{A}$ areas. Furthermore an area with a point in Q cannot be dominated among $\mathcal{A}$ as, by Lemma 6.7, this would make A_i also dominated instead of dependent among $\mathcal{A}$. Hence every area in $\mathcal{A}$ that contains a point in Q must be maximal among $\mathcal{A}$. Therefore A_j is maximal among $\mathcal{A}$.

In order to solve the problem we have to determine about area A_i, i.e. we have to perform updates until area A_i changes its status from dependent to either maximal or dominated among $\mathcal{A}$, as by Remark 4.5. We note that A_i can only change its status by updating A_i itself or an area which intersects with Q.

We first show the witness set for the special case where $A_j \cap \overline{Q} = \{h_i\}$. In this case we note that $h_i \in A_i$ as otherwise, by our construction of Q, we would have $h_i \in Q$ instead of $h_i \in \overline{Q}$. So we have that $h_i \in A_i \cap A_j$. As $h_i \in A_j$ and A_j is maximal among $\mathcal{A}$, there does not exist a point in any area of $\mathcal{A} \setminus \{A_j\}$ that is higher than h_i. Therefore no other area in $\mathcal{A} \setminus \{A_j\}$ can intersect with Q and we have the witness set $\{A_i, A_j\}$. See Figure 6.9.

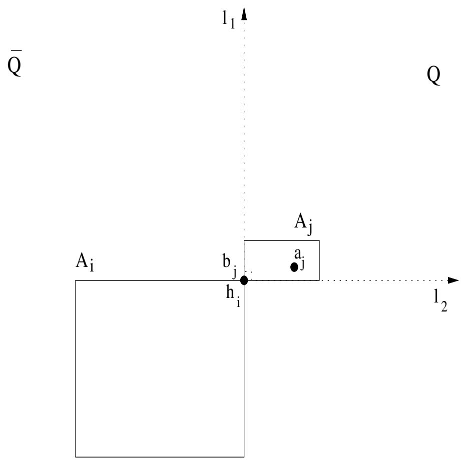


Figure 6.9: Case $A_j \cap \overline{Q} = \{h_i\}$

We now look at the case $A_j \cap \overline{Q} \neq \{h_i\}$. As $A_j \cap \overline{Q} \neq \{h_i\}$ then $A_j \cap \overline{Q}$ contains a point $c_j = (c_j^x, c_j^y)$ such that $c_j \neq h_i$. Therefore, by our construction of Q, either $c_j^x < h_i^x$ or $c_j^y < h_i^y$ and hence, by Definition 6.5, A_j is split by either l_1 or l_2. Without loss of generality lets assume $c_j^x < h_i^x$ and A_j is split by l_1. Further let A_k be another area with a point in Q. With the similar arguments for A_j we have that $A_k \cap Q \neq \emptyset$, $A_k \cap \overline{Q} \neq \emptyset$ and A_k contains a point $c_k = (c_k^x, c_k^y)$ such that $c_k \neq h_i$ and either $c_k^x < h_i^x$ or $c_k^y < h_i^y$. As A_j is maximal among $\mathcal{A}$ and split by l_1, by Lemma 6.8, then $c_k^y < h_i^y$ and A_k must be split by l_2. So no other area in $\mathcal{A} \setminus \{A_j, A_k\}$ can intersect with Q and in this case we have the witness set $\{A_i, A_j, A_k\}$. If A_k does not exist then at least A_j exists and $\{A_i, A_j\}$ is a witness set. See Figure 6.10.

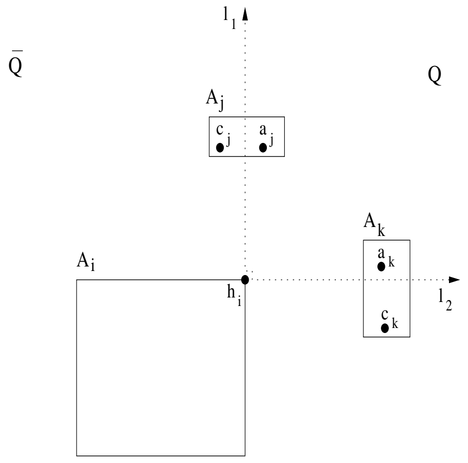


Figure 6.10: Case $A_j \cap \overline{Q} \neq \{h_i\}$

□

We note that the proofs of the above Lemmas required a somewhat different approach than the ones in [6] because of the different uncertainty area restrictions. For example in [6] it was stated that a line l splits an area A if $A - l$ is not connected, which we had to redefine in Definition 6.5 as the MPDP problem allows disconnected areas.

Following Lemma 6.6 and Lemma 6.9 we have:

Remark 6.10. *Algorithm 2 is 3-update competitive for the MPDP online problem.*

6.4.2 Lower Bound

We now argue that there does not exist a k-update competitive algorithm for $k < 3$ in Lemma 6.11. The reasoning of the proof is again taken from [6] which also applies for MPDP.

Lemma 6.11. *There exist configurations for the MPDP problem such that every deterministic online update strategy performs three times as many updates as needed.*

Proof. First we construct a gadget that contains three uncertainty areas A_1, A_2 and A_3 as shown in Figure 6.11. We can see that areas A_1 and A_3 are both maximal among $\mathcal{A}$, but updates are needed to determine about area A_2. Note that this figure demonstrates an instance with no partly among $\mathcal{A}$ areas, but with one dependent area (A_2) and is therefore an example of Lemma 6.9. We show with Figures 6.12, 6.13 and 6.14 that for any update strategy S there exists a configuration of precise location points for these three areas such that S requires three updates where only one was needed. Depending on the order in which S chooses to update areas, a precise location point $p_1 \in A_1$ or $p_2 \in A_2$ or $p_3 \in A_3$ is given as follows:

- If S updates A_1 first, we choose either Figure 6.12 or Figure 6.14 to be the input, and in both cases further updates are needed. If S updates A_2 next, we give Figure 6.12 to be the input. In this case S is forced to further update A_3, as area A_2 does not change its status to either maximal or dominated among $\mathcal{A}$ after the updates of areas A_1 and A_2. However the update of area A_3 on its own from Figure 6.12 is enough to change the status of area A_2 to dominated among $\mathcal{A}$. Similarly updating first area A_1 and then area A_3, Figure 6.14 is given to S to be the input, S must further update area A_2 and the single update of A_2 would have been enough to determine about A_2.

- If S updates A_2 first, we choose either Figure 6.12 or Figure 6.13 to be the input, and in both cases further updates are needed. If S updates A_1 next, we give Figure 6.12 to be the input. In this case S is forced to further update A_3, as area A_2 does not change its status to either maximal or dominated among $\mathcal{A}$ after the updates of areas A_2 and A_1. However the update of area A_3 on its own from Figure 6.12 is enough to change the status of area A_2 to dominated among $\mathcal{A}$. Similarly updating first area A_2 and then area A_3, Figure 6.13 is given to S to be the input, S must further update area A_1 and the single update of A_1 would have been enough to determine about A_2.

- If S updates A_3 first, we choose either Figure 6.13 or Figure 6.14 to be the input, and in both cases further updates are needed. If S updates A_1 next, we give Figure 6.14 to be the input. In this case S is forced to further update A_2, as area A_2 does not change its status to either maximal or dominated among $\mathcal{A}$ after the updates of areas A_3 and A_1. However the update of area A_2 on its own from Figure 6.14 is enough to change the status of area A_2 to dominated among $\mathcal{A}$. Similarly updating first area A_3 and then area A_2, Figure 6.13 is given to S to be the input, S must further update area A_1 and the single update of A_1 would have been enough to determine about A_2.

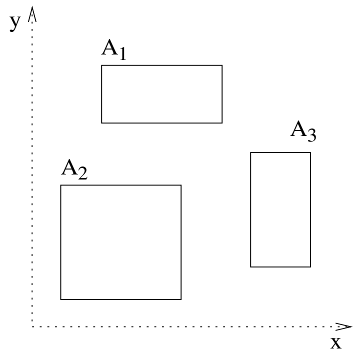
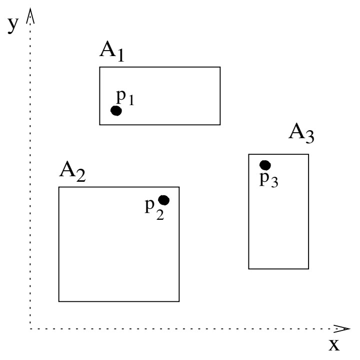
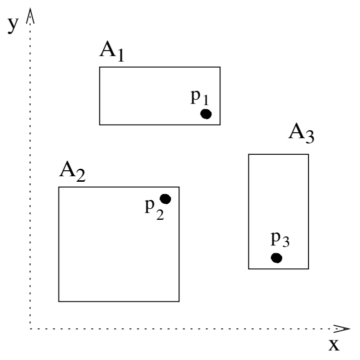


Figure 6.11: Lower Bound - Main Configuration





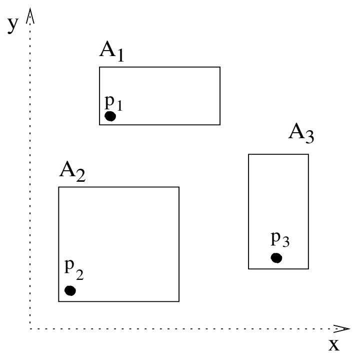


Figure 6.12: Config. 1 Figure 6.13: Config. 2 Figure 6.14: Config. 3

To show the lower bound we simply construct multiple gadgets where each is placed below and to the right of the previous one (see Figure 6.15). This guarantees that no area from one gadget contains a point higher than a point from an area of some other gadget. Hence, for each gadget, OPT performs 1 update and any deterministic online algorithm performs 3 updates. Therefore for j gadgets of this construction $OPT = j$ and any algorithm is $3j$. This concludes our proof that there does not exist a k-update competitive algorithm for the MPDP problem with $k < 3$.

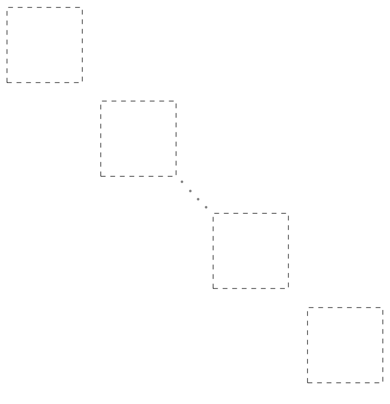


Figure 6.15: Placement of gadgets

□

6.4.3 Conclusion

Following Remark 6.10 and Lemma 6.11, we summarize our results for this section with Theorem 6.12, which is equivalent to Theorem 1 by Bruce et al. [6]:

Theorem 6.12. *The witness algorithm for the online problem of MPDP is 3-update competitive. Furthermore, this is the best possible.*

6.5 Verification Problem

6.5.1 Introduction

In this section we look at the verification setting of the maximal points problem under uncertainty, with the direct products restriction in place as proposed in Section 6.1. An instance of the Maximal Point Direct Products Verification problem (MPDPV for short) is the pair $(P, \mathcal{A})$, defined by combining Definition 4.2 (MPV) and Definition 6.1 (MPDP). In other words P is a set of points in the plane and $\mathcal{A}$ is a set of uncertainty areas for P, such that each uncertainty area consist only of points produced by the direct product of a set of x and a set of y coordinate values. The aim is to identify a minimal set of areas in $\mathcal{A}$, that when updated verifies the maximal points among P as maximal based on the information of $\mathcal{A}$ and the results of the updates (optimal update solution). As the precise locations are also directly given, in contrast with the online setting, we note that the algorithm can look up at the effect of including or not including a particular area to be part of an update solution.

We will present a polynomial time algorithm that finds a solution for MPDPV and argue about its correctness and time complexity. As shown earlier for Theorem 5.8 in Chapter 5 the MPV problem is NP-Hard, even if the uncertainty areas contain at most two points, and the proof for the NP-Hardness takes advantage of the disconnectivity of the areas. So it is interesting that them MPDPV problem allows disconnected areas and there exists a polynomial time algorithm for it.

The algorithm runs in two phases summarized as follows. In the first phase the algorithm collects and simulates updates under some criteria, which we will analyse later on, hence altering the set $\mathcal{A}$ in the process. In this phase the algorithm further collects a series of choices between updates, where all choices are such that every update solution would need to satisfy. The second phase then establishes a minimal set of areas that need to be updated in order to satisfy all the choices collected from the first phase. Thus, the set of updates performed in the first phase along with a minimal set of updates that satisfies all the choices, together form a solution for the MPDPV problem.

We will show that after Phase 2 of the MPDPV algorithm (Algorithm 4) has finished execution, each uncertainty area has status of either maximal or dominated among $\mathcal{A}$. Hence, by Remark 4.5, the set of all the updates performed is indeed an update solution. Furthermore this set of updates is of minimal size and therefore solves the MPDPV problem. Finally we show that all steps of the algorithm can be done in polynomial time and hence satisfy Theorem 6.38.

As defined earlier in Chapter 4 with Definition 4.4 the uncertainty areas are classified into maximal, partly, dependent and dominated among $\mathcal{A}$, as also, by Definition 4.1, their precise location point is classified as either maximal or non-maximal among P. Furthermore, as $\mathcal{A}$ changes during the process of the algorithm when updates are simulated, the classification of the uncertainty areas changes. To emphasize this we combine the two classifications into one, for example an uncertainty area A is **D-M** if A is dependent among the current set $\mathcal{A}$ and its precise location $p \in A$ is maximal among P. In the same way we have the following types for area A: **D-N** if A is dependent among $\mathcal{A}$ and p is non-maximal among P, **P-M** if A is partly among $\mathcal{A}$ and p is maximal among P, and **P-N** if A is partly among $\mathcal{A}$ and p is non-maximal among P. Note that there are also areas of type **M-M** (A maximal among $\mathcal{A}$, p maximal among P) and **N-N** (A dominated among $\mathcal{A}$, p non-maximal among P) which the algorithm does not deal with. This is because, as derived by Remark 4.5, when the current state of $\mathcal{A}$ consists only of areas that are either maximal or dominated among $\mathcal{A}$, no further updates are needed to solve a maximal point under uncertainty problem. Therefore we concentrate on 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 uncertainty area $A \in \mathcal{A}$ belongs to one of the four types D-M/D-N/P-M/P-N but it also has the following property which renders it essential to be included in every update solution. If $\mathcal{A}'$ is the set of uncertainty areas after updating every area in $\mathcal{A} \setminus \{A\}$ and there exists an area $B \in \mathcal{A}'$ which is neither maximal nor dominated among $\mathcal{A}'$ then A must be inside every update solution. Formally:

Definition 6.13. *Area A is self determined if $\mathcal{A} \setminus \{A\}$ is not an update solution.*

Lemma 6.14. *Let $A \in \mathcal{A}$ be an uncertainty area. If $\mathcal{A} \setminus \{A\}$ is not an update solution, then every update solution must contain A.*

Proof. This is quite easy to see, however we give a more detailed analysis.

Let $\mathcal{A} \setminus \{A\}$ not be an update solution and lets assume there exists an update solution that does not contain A.

By updating $\mathcal{A} \setminus \{A\}$ all areas except A become trivial. Area A cannot be trivial as well since then, after updating $\mathcal{A} \setminus \{A\}$, there would not be any uncertainty in

$\mathcal{A}$ and the set of all maximal points would obviously be computable; but by our assumption $\mathcal{A} \setminus \{A\}$ is not an update solution.

Since $\mathcal{A} \setminus \{A\}$ is not an update solution, by the reasoning of Remark 4.5, after updating $\mathcal{A} \setminus \{A\}$ there exists an area that is either partly or dependent among $\mathcal{A}$. So lets assume that $\mathcal{A} \setminus \{A\}$ have been updated.

No area in $\mathcal{A} \setminus \{A\}$ can be partly among $\mathcal{A}$ since, by Definition 6.2, a partly area contains at least two points and therefore only A can be partly. Area A cannot be dependent among $\mathcal{A}$ since there must be another area in $\mathcal{A} \setminus \{A\}$ containing a point higher and a point not higher than A, but they are all now trivial. So we have the two cases as follows:

Case 1. Area A is partly and hence A itself must be updated to change its status to either maximal or non-maximal among $\mathcal{A}$. Therefore there does not exist an update solution that does not contain A, contradiction.

Case 2. There exists area $B \in \mathcal{A} \setminus \{A\}$ which is dependent among $\mathcal{A}$. As B is dependent there exists an area in $\mathcal{A} \setminus \{B\}$ with a point higher and a point not higher than B (also no area currently in $\mathcal{A}$ is higher than B as it would be dominated instead of dependent among $\mathcal{A}$). Since A is now the only non-trivial area in $\mathcal{A}$ then A is the one with this property. So without updating A area B cannot change its status to either maximal or non-maximal among $\mathcal{A}$. Therefore there does not exist an update solution that does not contain A, contradiction.

$\square$

The outline of the algorithm is as follows:

Algorithm 3 MPDPV algorithm - Phase 1

```
1: Initialize: Problem instance $(P, \mathcal{A})$
2: Step 1. Update all self determined areas;
3: Classify all areas to D-M/D-N/P-M/P-N;
4: for all $i$ such that $A_i$ is a D-N area do
5:     if no other D-N area has precise location higher than $A_i$ then
6:         if there exists exactly one area $B$ with precise location higher than $A_i$ then
7:             Step 2. Update area $B$;
8:             Classify all areas to D-M/D-N/P-M/P-N;
9:         else (there exist exactly two areas $B$ and $C$ with precise location higher than $A_i$)
10:            Step 3. Record choice: $B$ or $C$;
11:        end if
12:    end if
13: end for
14: for all $i$ such that $A_i$ is a P-M area do
15:     if there exists exactly one area $B$ with a point higher than a point in $A_i$ then
16:         Step 4. Record choice: $A_i$ or $B$;
17:     else (there exist exactly two areas $B$ and $C$ that contain a point higher than a point in $A_i$)
18:         Step 5. Record choice: $A_i$ or $\{B, C\}$;
19:     end if
20: end for
    return The modified instance $(P, \mathcal{A}')$, the set of areas updated and the set of choices.
```

Algorithm 4 MPDPV algorithm - Phase 2

```
1: Initialize: From Phase 1 - Problem instance $(P, \mathcal{A}')$, set of areas updated and set of choices;
2: Step 1. Sort all areas found in the choices by the $x$ increasing order of their precise location;
3: Step 2. Link all recorded choices in chains;
4: Step 3. Update the second and every other area in every chain;
   return The set of all areas updated.
```

6.5.2 Algorithm Phase 1

Overview.

The algorithm first establishes a set of updates that have to be in every 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 performed and otherwise the choice between some updates is recorded.

So the first phase simulates some updates and hence $\mathcal{A}$ changes accordingly during the process. All these updates and choices, along with the updated instance of the problem, are then going to be used for Phase 2.

Step 1 - Self determined areas.

For the first step we establish a set of updates that has to be a subset of every update solution. Let $A \in \mathcal{A}$ be an area such that after updating every area in $\mathcal{A}$ except A does not solve the problem. Formally, $\mathcal{A} \setminus \{A\}$ is not an update solution and so any update solution must contain A as by Lemma 6.14. **The update of all such areas in this step is simulated by the algorithm**, and therefore the set $\mathcal{A}$ is now modified with the updated data. As by Lemma 6.14, and as an effect of the action performed we have Remark 6.15 and Lemma 6.16:

Remark 6.15. *All updates performed in Step 1 are needed in every update solution.*

Lemma 6.16. *After the updates of Step 1, $\mathcal{A}$ does not contain any D-M or any P-N areas.*

Proof. Let's assume there exists an area A_1 such that it is D-M after Step 1. As A_1 is dependent among $\mathcal{A}$, there exists an area $A_2 \in \mathcal{A} \setminus \{A_1\}$ with a point $q_2 \in A_2$ such that $q_2 > A_1$. As the precise location point $p_1 \in A_1$ is maximal among P, then the precise location $p_2 \in A_2$ is such that $p_2 \not> p_1$, and we note $p_2 \neq q_2$. Without updating A_2, though, area A_1 cannot be identified as maximal among $\mathcal{A}$ since q_2 remains in A_2 and $q_2 > A_1$. So $\mathcal{A} \setminus \{A_2\}$ is not an update solution and therefore A_2 must have been updated by Step 1. This is a contradiction since A_2 is non trivial as $q_2, p_2 \in A_2$ and $p_2 \neq q_2$.

Let's assume that after Step 1 there exists a P-N area A_1. As A_1 is partly among $\mathcal{A}$ it must contain a point h_1 such that no point in any other area of $\mathcal{A} \setminus \{A_1\}$ is higher than h_1. Furthermore the precise location $p_1 \in A_1$ is non-maximal among P and therefore $p_1 \neq h_1$. Without updating A_1 itself, this area cannot be identified as dominated among $\mathcal{A}$, as point h_1 would remain in A_1. So $\mathcal{A} \setminus \{A_1\}$ is not an update solution and therefore A_1 must have been updated by Step 1. This is a contradiction since A_1 is non trivial as $h_1, p_1 \in A_1$ and $p_1 \neq h_1$.

□

Steps 2 and 3 - D-N areas.

In the second and third steps we focus on dependent among $\mathcal{A}$ areas with precise location non-maximal among P (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_1 be a D-N area after Step 1 is performed. As A_1 is D-N then its precise location p_1 is not maximal among P and therefore there exists $p_2 \in P$ such that $p_2 > p_1$. Note that there may be another D-N area with precise location higher than A_1. We will only take action when this is not the case and we will show later on with Lemma 6.25 why this suffices for not leaving any D-N areas. At this stage there exist either one or two areas with precise location higher than A_1. Formally:

Lemma 6.17. *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 for the uncertainty areas. Area A_i is made by the direct product of a set A_i^x and a set A_i^y.

Let $h_i = (h^x, h^y)$ be the point such that h^x is the upper limit of A_i^x and h^y is the upper limit of A_i^y. We note point h_i may or may not be in A_i, depending on if both upper limits of A_i^x and A_i^y are inclusive or not, and h_i is higher than all other points in A_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 among 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 area A_j (the area of 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 6.16. So A_j is either an M-M, P-M or D-M area. In all cases point p_j is maximal among P.

As area A_i is dependent among $\mathcal{A}$, 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 $q_j = (p_j^x, q_j'^y)$ or $q_j = (q_j'^x, p_j^y)$ such that $q_j \not> A_i$ and $q_j < p_j$ (see Figure 6.16). 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. Figure 6.16 below shows these properties.

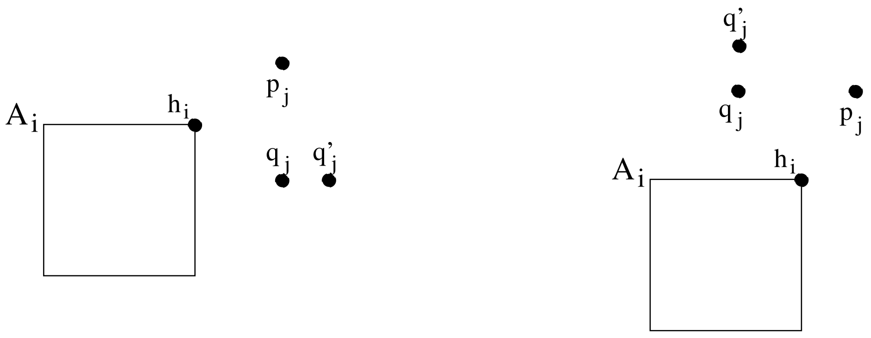


Figure 6.16: D-N area A_i with $q_j = (p^x_j, q'^y_j)$ or $q_j = (q'^x_j, p^y_j)$

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, point p_k is maximal among P and 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 $p^x_k \geq p^x_j \geq q^x_j$ or $p^x_j \geq p^x_k \geq q^x_k$. 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 $p_j > q_k$. Without loss of generality let $p_k > q_j$, see Figure 6.17. So area A_j contains a point that is dominated by a point in P from another area, while its precise location is maximal among P. Therefore without updating A_j itself, this area cannot change status to maximal among $\mathcal{A}$. In other words area A_j is self determined and would have been updated in Step 1 becoming trivial.

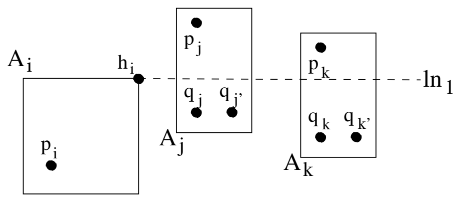


Figure 6.17: Example of D-N area A_i. Step 1 of the algorithm would update area A_j

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.

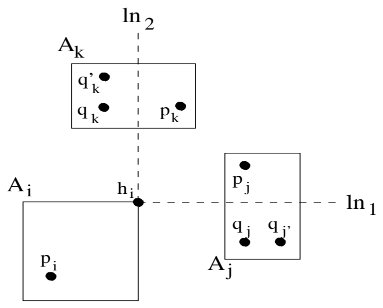


Figure 6.18: D-N area A_i after Step 1 of the algorithm where $p_j \in A_j$ and $p_k \in A_k$ such that $p_j, p_k > A_i$

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. Figure 6.18 shows the formation.

□

We now give the definition of neighbouring areas and note a property from the proof of Lemma 6.17 in Remark 6.19:

Definition 6.18. *Let A_1 and A_2 be two uncertainty areas with precise location points $p_1 \in A_1$ and $p_2 \in A_2$. Further let l be the line starting at p_1 and ending at p_2. For every point $q \in l$, if there does not exist a precise location point $p \in P \setminus \{p_1, p_2\}$ such that $p \geq q$ then we say A_1 and A_2 are neighbouring areas.*

Remark 6.19. *Let A_1 be a D-N area such that there does not exist another D-N area with precise location higher than A_1, but there exist two areas A_2 and A_3 with precise location higher than A_1. Then areas A_2 and A_3 are neighbouring areas and furthermore have precise locations maximal among P.*

Following Lemma 6.17, 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_1 be a D-N area such that there exists exactly one area A_2 with precise location p_2 such that $p_2 > A_1$. As A_1 is dependent among $\mathcal{A}$, then A_2 must contain at least one point that is not higher than A_1. As p_2 is the only point in P that is higher than A_1, without updating neither A_1 nor A_2, area A_2 remains dependent among $\mathcal{A}$. So any update solution must contain at least either A_1 or A_2. **The action the algorithm takes here is to update $\mathbf{A_2}$**, and therefore set $\mathcal{A}$ is now modified with the updated data.

The definition of a choice that will be recorded by Step 3,4 and 5 of the algorithm is as follows:

Definition 6.20. *A choice is a pair of sets of areas. We say a set of updates S satisfies a choice $C = (S_1, S_2)$ if S contains either S_1 or S_2. We distinguish between* *two types of choices: A choice where each set is of size one is called type A, and where one set has size one and the other size two is called type B.*

Step 3. Let A_1 be a D-N area such that there exist exactly two areas (namely A_2 and A_3) with precise location higher than A_1. Similar to the arguments in Step 2 every update solution must contain at least one of the three areas A_1, A_2 and A_3. At this stage no updates are simulated but instead **the algorithm records the choice between $\mathbf{A_2}$ and $\mathbf{A_3}$** (type A choice). We will use Lemma 6.21 and Lemma 6.22 to justify the actions taken for Step 2 and Step 3.

Lemma 6.21. *Let U be an update solution for an instance of the MPDPV problem with $A_1 \in U$, and let area A_1 be dominated among $\mathcal{A}$ after the updates of $U \setminus \{A_1\}$. Then $U \setminus \{A_1\}$ is also an update solution.*

Proof. Let $U \setminus \{A_1\}$ not be update solution and let this set of updates be performed on the instance of the problem. As this is not an update solution there must exist now an area A_2 that is neither maximal nor dominated among $\mathcal{A}$ (we note that it is other than A_1 as A_1 is dominated among $\mathcal{A}$). Also, as U is an update solution and $U \setminus \{A_1\}$ is not, the further update of A_1 must change the status of A_2 to either maximal or dominated among $\mathcal{A}$. Therefore there must exist a point $a_1 \in A_1$ and a point $a_2 \in A_2$ such that $a_1 > a_2$.

As A_1 is dominated among $\mathcal{A}$ and as $a_1 \in A_1$, $a_2 \in A_2$ with $a_1 > a_2$, then there exists an area $A_3 \in \mathcal{A} \setminus \{A_1, A_2\}$ such that $A_3 > A_1$ and therefore also $A_3 > a_1$. Furthermore it must follow that $A_3 > a_2$, since $A_3 > a_1 > a_2$. As such, area A_2 cannot change status to maximal among $\mathcal{A}$ after updating A_1, since $A_3 > a_1$ remains. Hence A_2 must change to dominated among $\mathcal{A}$ after updating A_1. Therefore the precise location point $p_1 \in A_1$ is higher than A_2 whereas $A_1 \not> A_2$.

As area A_1 is dominated among $\mathcal{A}$ with $p_1 \in A_1$ and $p_1 > A_2$ then, by Lemma 6.7, area A_2 must already be dominated among $\mathcal{A}$, before updating A_1. A contradiction. □

Lemma 6.22. *Let A_1 be a D-N area and let A_2 be another area with precise location point p_2 such that $p_2 > A_1$. If there exists a set of areas U that is an update solution with $A_1 \in U$ then $(U \bigcup \{A_2\}) \setminus \{A_1\}$ is also an update solution.*

Proof. Since U is an update solution the set $U \bigcup \{A_2\}$ is also an update solution. As $A_1 \in U$, it follows that $A_1 \in U \bigcup \{A_2\}$. Since $p_2 > A_1$ and p_2 is the precise location of A_2, after updating area A_2, area A_1 will have the status of dominated among $\mathcal{A}$. This is also the case after updating $(U \bigcup \{A_2\}) \setminus \{A_1\}$ since $A_2 \in (U \bigcup \{A_2\}) \setminus \{A_1\}$. Finally since $U \bigcup \{A_2\}$ is an update solution, $A_1 \in U \bigcup \{A_2\}$ and A_1 is dominated among $\mathcal{A}$ after updating $(U \bigcup \{A_2\}) \setminus \{A_1\}$, by Lemma 6.21, $(U \bigcup \{A_2\}) \setminus \{A_1\}$ is also an update solution.

□

For Step 2, as by Lemma 6.22, an update solution containing A_1 can be modified by replacing A_1 with A_2 with the resultant set still being an update solution. The size does not increase since we are replacing one area with another, and so the update of A_2 is not in every update solution but there must be an optimal update solution that contains A_2. For Step 3, as by Lemma 6.22, an update solution containing A_1 can be modified by replacing A_1 with either A_2 or A_3 with the resultant set still being an update solution. The size again does not increase and so there must be an optimal update solution that contains either A_2 or A_3. We give our conclusions for Step 2 and 3 in Remark 6.23, Remark 6.24 and Lemma 6.25:

Remark 6.23. *There exists an optimal update solution that contains the updates of Step 2.*

Remark 6.24. *Every update solution must satisfy all choices collected by Step 3.*

Lemma 6.25. *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_1 such that it is D-N after the updates of Step 2 and after the updates of a set that satisfies the choices of Step 3.

If A_1 contains a point a that is maximal among P then $\mathcal{A} \setminus \{A_1\}$ would not be an update solution and therefore, by Lemma 6.14, A_1 would be updated in Step 1, become trivial and no longer have a. So it does not contain a point that is maximal among P, and we now have two cases: either there does not exist another D-N area with precise location higher than A_1 (Case 1) or there exists one (Case 2).

Case 1: As by Lemma 6.17 there exists either one or two areas with precise location higher than A_1. If there is one then Step 2 would update an area A_2 with precise location p_2 such that $p_2 > A_1$ and therefore A_1 would change status to N-N. So there must exist two areas with precise location higher than A_1. 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_1, which again changes the status of A_1 to N-N. So A_1 cannot be dependent among $\mathcal{A}$, contradiction.

Case 2: Out of all D-N areas with precise location higher than A_1, there must exist an area A_2 with precise location p_2 such that for itself there does not exist a D-N areas with precise location higher than A_2. Hence A_2 falls to Case 1 and has changed status to dominated among $\mathcal{A}$. As area A_2 is dominated among $\mathcal{A}$ with $p_2 \in A_2$ and $p_2 > A_1$ then, by Lemma 6.7, area A_1 must be already be dominated instead of dependent among $\mathcal{A}$. A contradiction.

□

Steps 4 and 5 - P-M areas.

In the fourth and fifth steps the remaining partly among $\mathcal{A}$ areas with precise location maximal among P (P-M) are considered. After Step 3 for each P-M area A_i there exist either one or two areas that contain a point higher than a point in A_i, formally:

Lemma 6.26. *Let A_i be a P-M area. There exist either one or two areas that contain a point higher than a point in A_i.*

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

Let $l_i = (l^x, l^y)$ be the point such that l^x is the lower limit of A_i^x and l^y is the lower limit of A_i^y. We note point l_i may or may not be in A_i, depending on if both lower limits of A_i^x and A_i^y are inclusive or not, 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 among $\mathcal{A}$ there must exist an area A_j with a point $q'_j \in A_j$ such that $q'_j > q_i$. As each point in A_i is maximal among P, the precise location $p_j \in A_j$ is such that $p_j \not> q_i$. We now choose a point $q_j \in A_j$ with $q_j = (p_j^x, q_j'^y)$ if $q_i^y > p_j^y$, or with $q_j = (q_j'^x, p_j^y)$ if $q_i^y > p_j^y$, and therefore $q_j > q_i$ and $q_j > p_j$ (see Figure 6.19). 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.

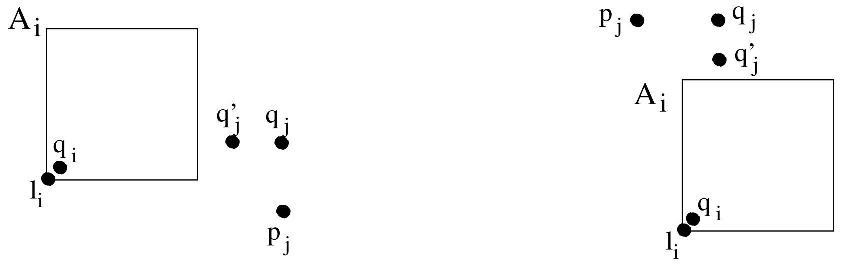


Figure 6.19: P-M area A_i with $q_j = (p_j^x, q_j'^y)$ or $q_j = (q_j'^x, p_j^y)$

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 6.25 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 6.16 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 point p_j is maximal among P.

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

Case $l_i \in A_i$: As $p_j \not> A_i$ then also $p_j \not> l_i$.

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 $p_j \not> A_i$, 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 $p_j \not> l_i$.

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 $p_k \not> A_i$.

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 $q_j > p_k$. Without loss of generality let $q_k > p_j$, see Figure 6.20. So although area A_j has its precise location maximal among P, without updating A_k, area A_j cannot change status to maximal among $\mathcal{A}$. In other words area A_k is self determined and would have been updated in Step 1 becoming trivial.

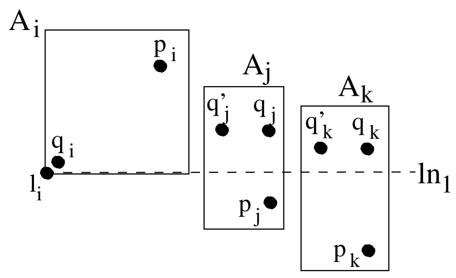


Figure 6.20: Example of P-M area A_i. Step 1 of the algorithm would update A_k

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.

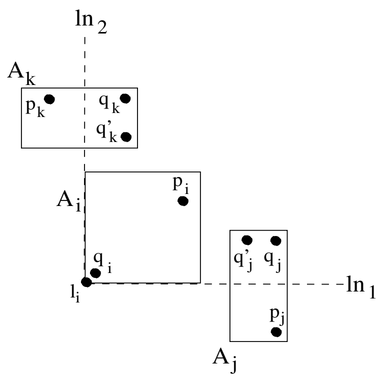


Figure 6.21: P-M area A_i after Step 1 of the algorithm where $q_j \in A_j$ and $q_k \in A_k$ such that $q_j, q_k > q_i$

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. Figure 6.21 shows the formation.

□

By Lemma 6.26, at this point, if A is a P-M area there exist either one or two areas that contain a point higher than a point in A. Furthermore the precise locations of those areas are not higher than any point in A (as $\mathcal{A} \setminus A$ would not be an update solution and A would be updated by Step 1). Again we separate these cases accordingly (Step 4 for one area and Step 5 for two areas.)

Step 4. Let A_1 be a P-M area such that there exists exactly one area, namely A_2, containing a point higher than a point in A_1. As the precise location of A_2 is not higher than any point in A_1, updating A_2 will change the status of A_1 to maximal among $\mathcal{A}$. Furthermore by updating A_1 this is also the case as there is no point higher than the precise location of A_1. As A_2 contains at least one point higher than a point in A_1, without updating neither A_1 nor A_2, area A_1 will remain partly among $\mathcal{A}$ and hence every 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 $\mathbf{A_1}$ and $\mathbf{A_2}$** (type A choice).

Step 5. We now consider two areas, namely A_2 and A_3, where each contains a point higher than a point of the P-M area A_1. With a similar reasoning as in Step 4 we have that without updating neither A_1 nor both A_2 and A_3, area A_1 remains partly among $\mathcal{A}$ and so every update solution must contain at least either A_1 or both A_2 and A_3. Again an update cannot be simulated and therefore the **algorithm records the choice between area $\mathbf{A_1}$ and the set of areas $\{\mathbf{A_2}, \mathbf{A_3}\}$** (type B choice).

With the collection of all choices of Step 4 and 5 and the proof of Lemma 6.26 we have the following properties in Remark 6.27, Remark 6.28 and Lemma 6.29:

Remark 6.27. *Let A_1 be a P-M area. By Lemma 6.26 either there exists one (namely A_2) or two (namely A_2 and A_3) areas which contain a point higher than a point in A_1. The precise locations of A_2 and A_3 are maximal among P. Further:*

- *If there exists one then A_1 and A_2 are neighbouring areas.*
- *If there exist two then A_1 is the middle neighbouring area of A_2 and A_3*

Remark 6.28. *Every update solution must satisfy all choices of Step 4 and Step 5.*

Lemma 6.29. *A set of updates that satisfies all choices collected by Step 4 and Step 5, does not leave any P-M areas.*

Proof. Let's assume there exists an area $A_1 \in \mathcal{A}$ such that A_1 is P-M after satisfying the choices of Step 4 and 5.

As A_1 is partly among $\mathcal{A}$, it contains a point h_1 such that no point in any area of $\mathcal{A}$ is higher than h_1. Area A_1 further contains a point l_1 such that some other area $A_2 \in \mathcal{A} \setminus \{A_1\}$ contains a point l_2 with $l_2 > l_1$ (as otherwise A_1 would be maximal instead of partly among $\mathcal{A}$). Therefore $l_1 \neq h_1$ and hence A_1 is not trivial.

We note that the precise location $p_2 \in A_2$ cannot be higher than any point in A_1, as otherwise $\mathcal{A} \setminus \{A_1\}$ would not be an update solution (since h_1 is maximal among P) and therefore A_1 would have been updated in Step 1 and become trivial. This follows that $p_2 \not> l_1$ and as also $l_2 > l_1$ then $l_2 \neq p_2$. Therefore A_2 is not trivial either.

As by Step 4 and 5, either A_1 itself must have been updated to satisfy the choice or all areas with a point higher than a point in A_1 must have been updated. As both A_1 and A_2 are not trivial then neither has been updated and therefore the choice was not satisfied, contradiction.

□

Having analysed all the steps of Phase 1, we now combine the results to show the desired outcome in Lemma 6.30 and Lemma 6.31:

Lemma 6.30. *All updates made in Phase 1 together with a set of updates that satisfies the choices collected, is an update solution.*

Proof. 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 can no longer be a type for an uncertainty area. Specifically: By Lemma 6.16 after Step 1 there are no D-M and no P-N areas in $\mathcal{A}$. By Lemma 6.25 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. Finally by Lemma 6.29 a set of updates that satisfies all choices of Step 4 and Step 5 does not leave any P-M areas. Therefore only the types M-M and N-N (i.e. either maximal or non-maximal among $\mathcal{A}$) are left, as required for an update solution stated in Remark 4.5.

□

Lemma 6.31. *All updates made in Phase 1 together with a minimal set of updates that satisfies all the choices, is an optimal update solution.*

Proof. In Lemma 6.30 we have shown that these update indeed form an update solution and we now argue about an optimal update solution.

At actions which perform updates it has been shown that either the updates must be in every update solution (Step 1 by Remark 6.15) or the updates must be in at least one optimal update solution (Step 2 by Remark 6.23). Hence the set of updates so far is minimal and let U_1 be this set.

By Remark 6.24 every update solution must satisfy all choices collected by Step 3, and by Remark 6.28 every update solution must satisfy all choices collected by Step 4 and Step 5. Let U_2 be a set of updates that satisfies all choices collected by Step 3, Step 4 and Step 5.

So if U_2 is of minimal size then, as U_1 is also minimal, so must be $U_1 \cup U_2$ and hence this is an optimal update solution.

□

We further give the time complexity for the procedure of classifying all uncertainty areas, before we argue about the overall time complexity of Phase 1.

Classifying uncertainty areas to D-M/D-N/P-M/P-N. To classify an uncertainty area to one of D-M, D-N, P-M and P-N we have to look at its status among $\mathcal{A}$ and the status of its precise location among P separately. We have given details about the process for both in Section 6.3. By Remark 6.3 the status among P takes polynomial time and by Remark 6.4 the status among $\mathcal{A}$ takes polynomial time. As both steps of the this process can be done in polynomial time then we note:

Remark 6.32. *Classifying uncertainty areas to D-M/D-N/P-M/P-N is done in polynomial time.*

Finally in this subsection we show the time complexity for Phase 1:

Lemma 6.33. *Phase 1 runs in polynomial time.*

Proof. The time complexity for the various steps of Phase 1 - Algorithm 3 is as follows:

Classifying areas. By Remark 6.32 the process of classifying uncertainty areas to D-M/D-N/P-M/P-N is done in polynomial time. In the algorithm this takes place once right after Step 1, and furthermore every time after Step 2 occurs. Therefore this process is done in polynomial time.

Step 1. For Step 1 the algorithm performs updates in case for each uncertainty area $A_i \in \mathcal{A}$ the set $\mathcal{A} \setminus \{A_i\}$ is not an update solution. So it compares A_i, with precise location point p_i, against all $p_j \in P \setminus p_i$. This is done in a similar way as the process of classifying areas, that is find the lowest point $l_i \in A_i$ and the highest point $h_i \in A_i$, determine whether they are inclusive and compare them against point p_j. For example, if the limits are inclusive, there is a match if $p_j \not> h_i$ and $p_j > l_i$. Therefore this process is done in polynomial time.

Step 2 and Step 3. We have a particular D-N area A_1 with precise location $p_1 \in A_1$. The algorithm needs to find the one or two areas which have their precise location point higher than A_1. This is done in a similar way as the process of classifying areas, that is find the highest point $h_1 \in A_1$, determine whether it is inclusive and compare it against each point $p_i \in P \setminus \{p_1\}$. For example, if the limit is inclusive, there is a match if $p_i > h_1$. Therefore this process is done in polynomial time.

Step 4 and Step 5. We have a particular P-M area A_1. The algorithm needs to find the one or two areas which contain a point higher than a point in A_1. This is done in a similar way as the process of classifying areas, that is find the lowest point $l_1 \in A_1$ and for each $A_i \in \mathcal{A} \setminus \{A_1\}$ find the highest point $h_i \in A_i$, determine whether they are inclusive and compare h_i against l_1. For example, if the limits are inclusive, there is a match if $l_i > h_1$. Therefore this process is done in polynomial time.

As we have shown that each step runs in polynomial time, so must Phase 1 overall.

□

6.5.3 Algorithm 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 so now in Phase 2, in order to produce an optimal update solution, we compute a minimal set of updates to satisfy all the recorded choices. We give a summary for the properties of choices from the previous subsection:

Remark 6.34. *Choices from Phase 1 are such that:*

- *They are between two sets of areas and either of type A, where each set is of size one, or of type B, where one set is of size one and the other of size two.*
- *Choosing neither of the two sets of a choice does not satisfy the choice.*

- *Choosing either of the two sets of a choice is enough to satisfy the choice.*
- *All have to be satisfied in order to produce an update solution.*
- *By Remark 6.19 and Remark 6.27*
  - *In a type A choice the two areas involved are neighbouring.*
  - *In a type B choice the singleton is the middle neighbouring area of the two areas in the other set.*
  - *Each area within each choice has precise location maximal among P.*

We define the following to link the choices together. We say two choices *overlap* if they share an uncertainty area and we further say a *chain* is a sequence of choices $\{C_1, \ldots, C_k\}$ if C_i overlaps with C_{i+1} for every $1 \leq i \leq k-1$ and the sequence is of maximal length.

The three steps of Phase 2, as seen in Algorithm 4, are as follows:

Step 1. The first step is to sort all areas in the choices by the x value of their precise location in increasing order.

Step 2. The second step is to link the choices into chains. In Figure 6.22 we show an example of a chain with the various types of choices recorded by steps of Phase 1.

Step 3. The third and final step of Phase 2 is to update every other area in every chain starting with the second.

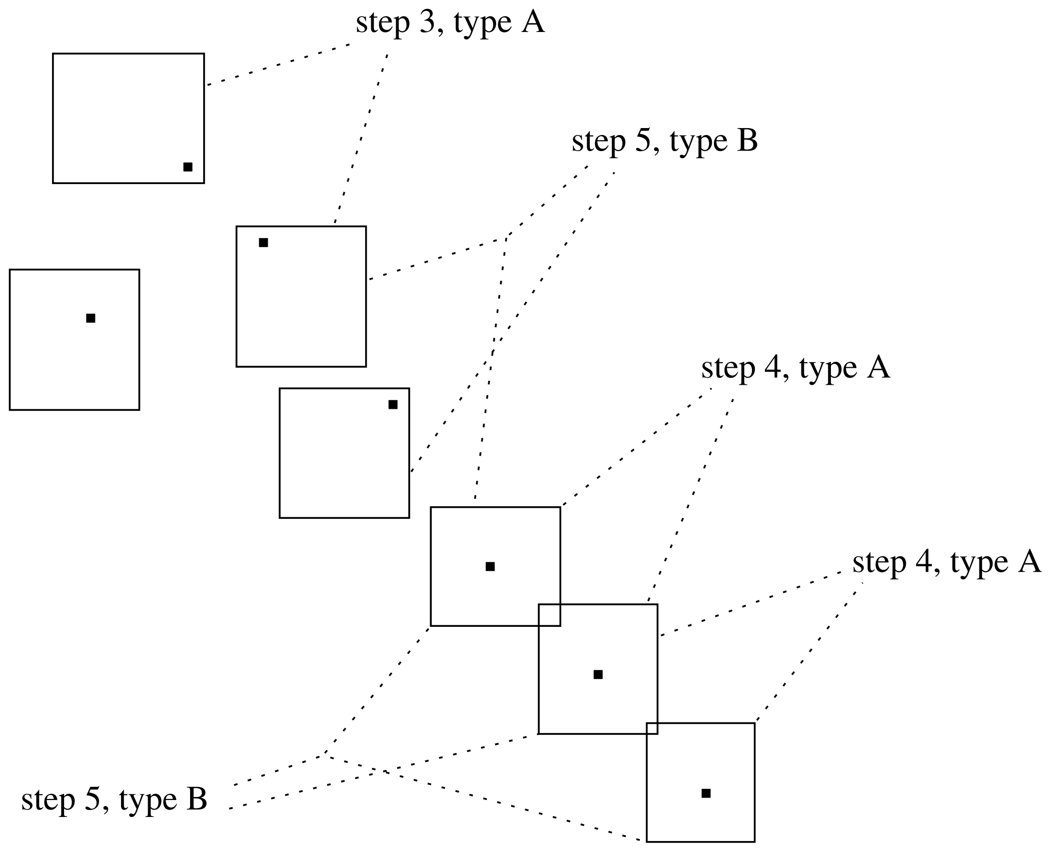


Figure 6.22: Example of a chain of choices

Assuming Step 1 and Step 2 of Phase 2 have been completed, we give the following properties for the resultant chains of choices in Lemma 6.35 and Lemma 6.36, and thus justify the action taken in Step 3:

Lemma 6.35. *By omitting two consecutive areas in a chain, there exists a choice which remains unsatisfied.*

Proof. As by Remark 6.19 and 6.27, the areas in a choice are neighbouring and their precise locations are maximal among P. Choices are between two (type A) or three (type B) areas. Furthermore for type B the choice can only be between the middle area and its two neighbouring side areas.

As such, by sorting these areas (for example by increasing order of the x value of their precise locations) and then linking them into chains, effectively any two consecutive areas in the same chain are shared in at least one choice together. So if two consecutive areas form a type A choice, then clearly by omitting both of them a choice remains unsatisfied. If two consecutive areas are part of a type B choice, then that means that one of them is the middle area in the choice and the other is either the left or right area of the choice. Therefore, again, by omitting both of them a choice remains unsatisfied. □

Lemma 6.36. *By choosing every other area in a chain starting with the second, all choices in the chain are satisfied with a minimal number of areas.*

Proof. By Lemma 6.35, we cannot omit two consecutive areas of a chain. So every update solution must contain at least all odd or at least all even areas of each chain.

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 as it is between two areas that are found consecutively in the chain. For a choice of type B choosing the odd or the even areas again satisfies the choice as it is between a middle and its two side areas. Hence, by choosing each even area in a chain, all choices in the chain are satisfied.

As each chain must contain at least two areas (i.e. at least one choice of type A), choosing all even areas in a chain is also minimal.

□

We further show the time complexity of Phase 2:

Lemma 6.37. *Phase 2 runs in polynomial time.*

Proof. The time complexity for Phase 2 - Algorithm 4 is as follows. First the the areas are are sorted, for example, by increasing order of the x value of their precise locations. Each choice is then compared against the next one to decide if the next one should be linked in the same chain or to a new chain. 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 other area in a chain starting with the second, takes at most $n/2$ iterations. So Phase 2 runs in polynomial time. □

By combining Lemma 6.31 and Lemma 6.36, the algorithm produces an optimal update solution. Furthermore, by Lemma 6.33 and Lemma 6.37 the whole process takes polynomial time and hence Theorem 6.38 is satisfied.

Theorem 6.38. *There exists a polynomial time algorithm that solves the MPDPV problem.*

Chapter 7

Maximal Points - Coordinate Specific Updates

7.1 Introduction

In this chapter we look at the maximal point under uncertainty problem, still considering only 2-dimensional points, where the update operations are performed slightly differently. Here the updates do not obtain the precise location of a point but instead updates on an uncertainty area are done separately for each of its coordinates. We use the same basic notation for maximal points under uncertainty as in Chapter 4, with the difference that an update does not necessarily reduce all the uncertain information of an area to trivial, but instead it reduces the uncertain information on the specified coordinate, in this case either x or y. So effectively one update as performed concerning maximal points in Chapter 4, 5 and 6 here corresponds to two updates. We look at the problem under the same restriction as by Definition 6.1 in Chapter 6, that is the areas of uncertainty are being restricted to be the direct product of a set of x and a set of y coordinate values.

Motivation. This setting of coordinate specific updates can be seen as a *refinement update* operation model. These models are motivated by scenarios of uncertainty problems where, given an uncertainty area for each input item, it is possible to update only part of it instead of obtaining the precise measurement. This may be more suitable if, for example, the cost of an update is dependent on the amount of uncertainty it reduces.

Overview. We will give results for the online problem showing the upper bound in Subsection 7.2.1 and a matching lower bound in Subsection 7.2.2 (following similar techniques as in Subsection 6.4.1 and 6.4.2 respectively), which then brings us to Theorem 7.8 in Subsection 7.2.3. We further briefly look at the verification problem is Section 7.3 where we show why our previous algorithm of Section 6.5 does not trivially carry over for coordinate specific updates. We denote the Maximal Point problem with this Coordinate Specific updates setting by MPCS, and for the Verification problem by MPCSV.

Update operation. Recall $(P, \mathcal{A})$ is an instance of a maximal points under uncertainty problem where $\mathcal{A}$ is the set of uncertainty areas and P is the set of precise locations. Following Definition 6.1 each uncertainty area $A_i \in \mathcal{A}$ is produced by the direct product of a set of x and a set of y coordinate values. Let these sets for A_i be A_i^x and A_i^y respectively, in other words $A_i = \{A_i^x \times A_i^y\}$, and let the precise location $p_i \in P$ of A_i be $p_i = (p_i^x, p_i^y)$. For A_i there is the option of updating either A_i^x or A_i^y. So the effect of updating A_i^x is that area A_i is altered such that $A_i = \{p_i^x \times A_i^y\}$. Similarly, after updating A_i^y, area A_i is altered such that $A_i = \{A_i^x \times p_i^y\}$. Finally after updating both A_i^x and A_i^y, area A_i becomes trivial with $A_i = \{p_i\}$.

7.2 Online Problem

7.2.1 Upper Bound

Overview. In this subsection we use the material as in Subsection 6.4.1, that is the witness algorithm for the maximal points under uncertainty problem (given in Algorithm 5) and a series of lemmas used to establish its update competitiveness. To some extend the proofs for the lemmas follow trivially from 6.4.1 and we show how to extend their use for the coordinate specific updates setting. Lemma 7.2 and 7.5 are used to establish witness sets for the two cases of Algorithm 5 (recall from Chapter 2 a witness set is a set of updates such that every update solution must contain at least one of those). Furthermore Lemma 7.3 and Lemma 7.4 are used to help the proof of Lemma 7.5. Moreover we give again Definition 7.1 for when a horizontal or vertical line splits an area which is being used for Lemma 7.4 and consequently Lemma 7.5. Since in the MPCS model two different updates on the same area correspond to one made on the MPDP model in Subsection 6.4.1, where we presented a 3-update competitive strategy, one can deduce that the same witness algorithm can be slightly modified and applied to MPCS producing a 6-update competitive algorithm. We will use similar reasoning and show that actually a 4-update competitive algorithm is possible.

We begin with restating the witness algorithm:

Algorithm 5 The witness algorithm for MPCS

```
1: while there exists at least one partly or one dependent area among $\mathcal{A}$ do
2:     if there exists a partly area among $\mathcal{A}$ then
3:         find a witness set $W$ ; update all areas in $W$
4:     else(there must exists a dependent area among $\mathcal{A}$)
5:         find a witness set $W$ ; update all areas in $W$
6:     end if
7: end while
```

The split in these two cases helps to identify witness sets. Note that the idea is to concentrate first on areas that are partly among $\mathcal{A}$ and witness sets concerning these areas. Only if there are such areas left in the given instance, will a strategy based on the existence of dependent among $\mathcal{A}$ areas be used to find witness sets.

We now simply restate the definition for when a horizontal or vertical line splits an uncertainty area, which will be used later on for the proof of Lemma 7.5:

Definition 7.1. *Let l_1 be a vertical line with equation $x = n$ and let l_2 be a horizontal line with equation $y = m$. Further let A be an uncertainty area. If there exist two points $a = (a_x, a_y)$ and $b = (b_x, b_y)$ in A such that $a_x \geq n > b_x$ then point b exist also with $a_y = b_y$ (due to the direct product restriction as in Definition 6.1) and we say l_1 splits A. Similarly if $a_y \geq m > b_y$ then also $a_x = b_x$ and we say l_2 splits A.*

If there exists an area that is partly among $\mathcal{A}$ for MPDP then as by Lemma 6.6 there exists a witness set of size at most 2. We extend this to Lemma 7.2 for MPCS:

Lemma 7.2. *Let $A = \{A^x \times A^y\}$ be a partly among $\mathcal{A}$ area, then there exists a witness set of size at most 4.*

Proof. Since A is partly among $\mathcal{A}$, it must contain points h and l such that: there does not exist a point in any area of $\mathcal{A} \setminus \{A\}$ that is higher than h; there exists at least one point b in an area $B \in \mathcal{A} \setminus \{A\}$ with $b > l$. Furthermore, potentially point $l \in A$ and area B exist such that $B > l$. If instead $B \not> l$ then at least $b > l$ with $b \in B$. We separate the cases for $B > l$ (Case 1) and $b > l$ (Case 2):

Case 1: $B > l$. By not updating both A^x and A^y, both points h and l remain in A. As $B > l$ then even after updating area B, the condition $B > l$ remains. We also know that there is no point in any area of $\mathcal{A} \setminus \{A\}$ that is higher than point h. Therefore without updating A itself this area cannot change its status to either maximal or dominated among $\mathcal{A}$, as needed for an update solution by Remark 4.5. Hence $\{A^x, A^y\}$ is a witness set. See Figure 7.1.

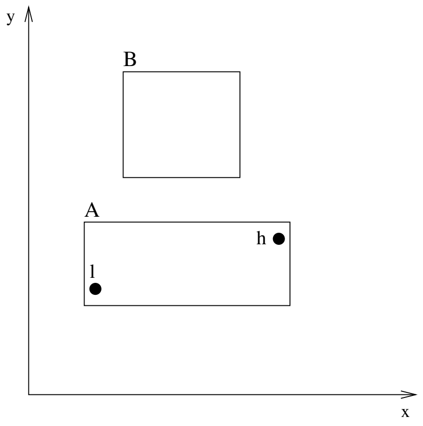


Figure 7.1: Case $B > l$

Case 2: $B \not> l$ but still $b > l$. With a similar reasoning as for Case 1, by only performing updates in areas of $\mathcal{A} \setminus \{A, B\}$, the points $b \in B$ and $l \in A$ remain such that $b > l$. Therefore, as point $h \in A$ also remains, at least one of $\{A^x, A^y, B^x, B^y\}$ has to be updated to change the status of area A to either maximal or dominated among $\mathcal{A}$. Hence $\{A^x, A^y, B^x, B^y\}$ is a witness set. See Figure 7.2.

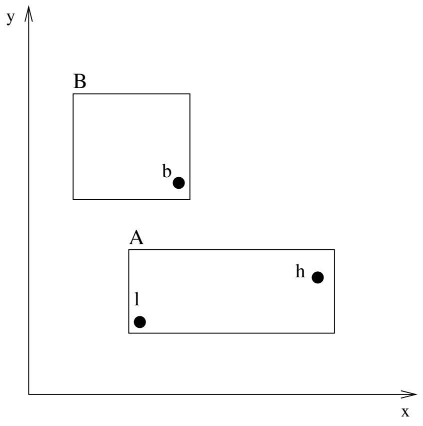


Figure 7.2: Case $b > l$

□

We recall a general property for dominated among $\mathcal{A}$ areas in Lemma 7.3 which we will use later on:

Lemma 7.3. *Let area A be dominated among $\mathcal{A}$ and let B be another area. If there exists a point $a \in A$ such that $a > B$ then area B is also dominated among $\mathcal{A}$.*

Proof. Same as for Lemma 6.7. □

We also recall a property about uncertainty areas produced from a direct product in Lemma 7.4:

Lemma 7.4. *A horizontal or vertical line can split at most one area that is maximal among $\mathcal{A}$.*

Proof. Same as for Lemma 6.8. □

If there are no partly among $\mathcal{A}$ areas but there exists a dependent among $\mathcal{A}$ area then as by Lemma 6.9 there exists a witness set of size at most 3 for MPDP. We extend this to Lemma 7.5 for MPCS:

Lemma 7.5. *If there are no partly areas but there exists a dependent area among $\mathcal{A}$, then there exists a witness set of size at most* 4.

Proof. Assume there are no partly areas among $\mathcal{A}$ and let $A_i = \{A_i^x \times A_i^y\}$ be a dependent area among $\mathcal{A}$ such that no other dependent area contains a point higher than A_i. Further let $h_i = (h_i^x, h_i^y)$ be the point made by the upper limits of the coordinate sets that make the direct product for A_i. We note that either $h_i \in A_i$ (upper limits for both x and y are closed) or $h_i \notin A_i$ (upper limit for either x or y is open). 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 these lines. 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 Q contains all the points that are higher than A_i. Also let $\overline{Q}$ denote the complement of Q. We show later in Figure 7.3 examples. The bounds of Q (lines l_1 and l_2) are shown with dotted lines, and non-inclusive limits of an uncertainty area are shown with dashed lines.

Since A_i is dependent among $\mathcal{A}$, there exists some other area $A_j = \{A_j^x \times A_j^y\}$ in $\mathcal{A}$ with a point $a_j \in A_j$ such that $a_j > A_i$. If this was not the case, then A_i would be partly among $\mathcal{A}$ as it would contain a point for which there does not exist a point in some other area such that is higher. As all points that are higher than A_i are inside Q, then $a_j \in Q$ and hence $A_j \cap Q \neq \emptyset$.

Furthermore there must exist another point $b_j \in A_j$ such that $b_j \not> A_i$, as otherwise $A_j > A_i$ and therefore A_i would be dominated instead of dependent among $\mathcal{A}$. Hence also $A_j \cap \overline{Q} \neq \emptyset$.

We show examples of A_i and A_j in Figure 7.3:

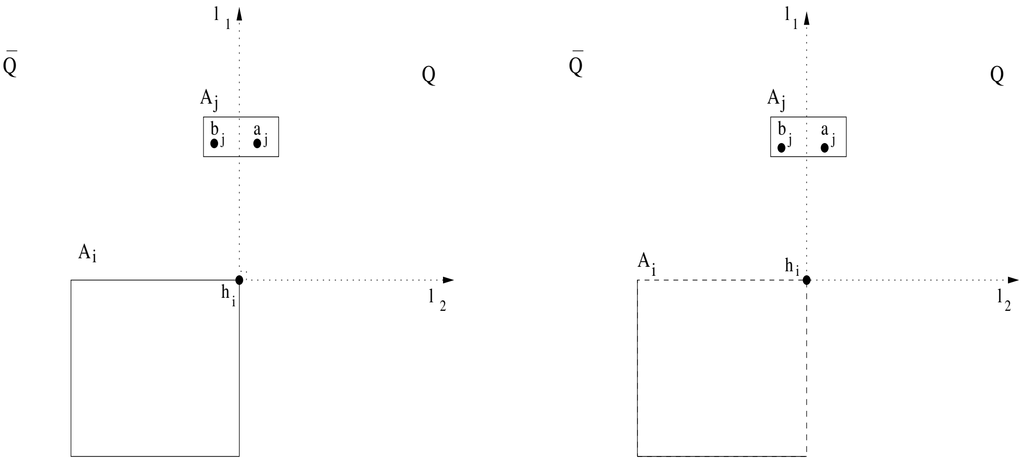


Figure 7.3: Construction of Q with either $h_i \in A_i$ or $h_i \notin A_i$

By our assumptions there are no other dependent among $\mathcal{A}$ areas with a point higher than A_i, and also no partly among $\mathcal{A}$ areas. Furthermore an area with a point in Q cannot be dominated among $\mathcal{A}$ as, by Lemma 7.3, this would make A_i also dominated instead of dependent among $\mathcal{A}$. Hence every area in $\mathcal{A}$ that contains a point in Q must be maximal among $\mathcal{A}$. Therefore A_j is maximal among $\mathcal{A}$.

In order to solve the problem we have to determine about area A_i, i.e. we have to perform updates until area A_i changes its status from dependent to either maximal or dominated among $\mathcal{A}$, as by Remark 4.5. We note that A_i can only change its status by performing updates that alter A_i itself or an area which intersects with Q.

We first show the witness set for the special case where $A_j \cap \overline{Q} = \{h_i\}$. In this case we note that $h_i \in A_i$ as otherwise, by our construction of Q, we would have $h_i \in Q$ instead of $h_i \in \overline{Q}$. So we have that $h_i \in A_i \cap A_j$. As $h_i \in A_j$ and A_j is maximal among $\mathcal{A}$, there does not exist a point in any area of $\mathcal{A} \setminus \{A_j\}$ that is higher than h_i. Therefore no other area in $\mathcal{A} \setminus \{A_j\}$ can intersect with Q and we have the witness set $\{A_i^x, A_i^y, A_j^x, A_j^y\}$. See Figure 7.4.

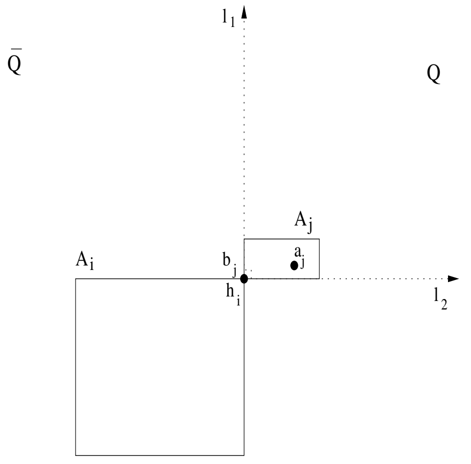


Figure 7.4: Case $A_j \cap \overline{Q} = \{h_i\}$

We now look at the case $A_j \cap \overline{Q} \neq \{h_i\}$. As $A_j \cap \overline{Q} \neq \{h_i\}$ then $A_j \cap \overline{Q}$ contains a point $c_j = (c_j^x, c_j^y)$ such that $c_j \neq h_i$. Therefore, by our construction of Q, either $c_j^x < h_i^x$ or $c_j^y < h_i^y$ and hence, by Definition 7.1, A_j is split by either l_1 or l_2. Without loss of generality lets assume $c_j^x < h_i^x$ and A_j is split by l_1. Further let $A_k = \{A_k^x \times A_k^y\}$ be another area with a point in Q. With the similar arguments for A_j we have that $A_k \cap Q \neq \emptyset$, $A_k \cap \overline{Q} \neq \emptyset$ and A_k contains a point $c_k = (c_k^x, c_k^y)$ such that $c_k \neq h_i$ and either $c_k^x < h_i^x$ or $c_k^y < h_i^y$. As A_j is maximal among $\mathcal{A}$ and split by l_1, by Lemma 7.4, then $c_k^y < h_i^y$ and A_k must be split by l_2. So no other area in $\mathcal{A} \setminus \{A_j, A_k\}$ can intersect with Q and we split the witness set cases for when there exists one area (A_j) and when there are two areas (A_j and A_k) which intersect with Q. If only A_j exists then $\{A_i^x, A_i^y, A_j^x, A_j^y\}$ is a witness set. If both A_j and A_k exist (assuming that $c_j \in A_j$ and $c_k \in A_k$ with $c_j^x < h_i^x$ and $c_k^y < h_i^y$), then $\{A_i^x, A_i^y, A_j^x, A_k^y\}$ is a witness set. The argument for not including A_j^y in the witness set is that for every point in A_j the y coordinate value must be greater than the y coordinate of every point in A_i (as otherwise A_j would also be split by l_2 and therefore, by Lemma 7.4, A_k would not exist) and hence whatever the update of A_j^y is cannot change the status of area A_i. With the similar reasoning but for the x coordinate we discard the update of A_k^x from the witness set. See Figure 7.5.

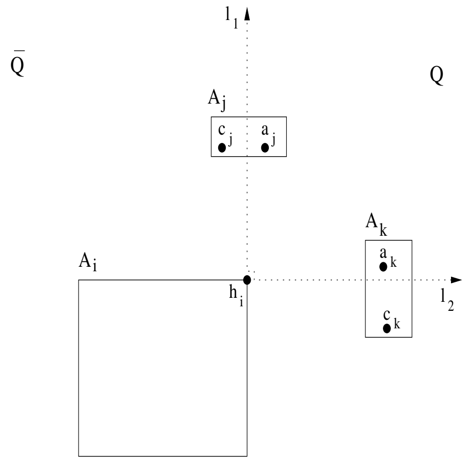


Figure 7.5: Case $A_j \cap \overline{Q} \neq \{h_i\}$

As all witness sets are at most size 4 this concludes our proof.

□

Following Lemma 7.2 and Lemma 7.5:

Remark 7.6. *Algorithm 5 is 4-update competitive for the MPCS online problem.*

7.2.2 Lower Bound

We now argue that there does not exist a k-update competitive algorithm with $k < 4$ for the MPCS problem:

Lemma 7.7. *There exist configurations for the MPCS problem such that every deterministic online algorithm performs four times as many updates as needed.*

Proof. We first give the construction of a gadget which consists of three uncertainty areas where updates are needed to solve the problem. The uncertainty areas are placed in such a way that, no matter in which order they are chosen to be updated by an update strategy, there exist configurations where 4 updates are made until the problem is solved. In addition to that, these configurations are such that the fourth update performed would by itself be enough to solve the problem. Then we conclude our reasoning by showing that this process can be repeated for an arbitrary number of times, without having an impact on the properties of each gadget.

Each gadget consists of three uncertainty areas denoted by A, B and C. See Figure 7.6.

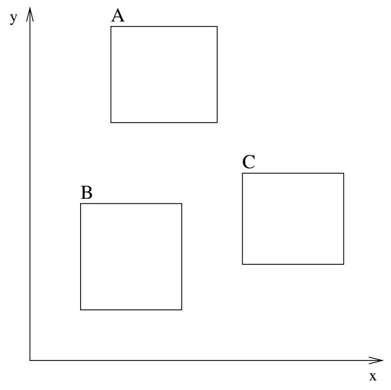


Figure 7.6: Areas A,B and C in the gadget

The uncertainty areas A and C are maximal among $\mathcal{A}$ and area B is dependent among $\mathcal{A}$. Furthermore both areas A and C contain points higher and not higher than B. As such no updates are needed to determine about A and C but updates are needed to determine about area B. While there are six different updates that can be performed in this instance (A^x,A^y,B^x,B^y,C^x and C^y), we note that updating A^y or C^x does not influence the status of area B. This is because even the lowest values of A^y and C^x are still higher than the highest value of B^y and B^x respectively. Also, as all uncertainty areas are the direct product of possible x and y values, updating a coordinate of an area does not reduce the possible values of the other coordinate of that area. So we will only consider performing updates out of the set $U = \{A^x, B^x, B^y, C^y\}$. We will use the Figures 7.7, 7.8, 7.9, 7.10 and 7.11 to illustrate how the updates are given to the update strategy. The dashed horizontal and vertical lines in the figures show what the uncertainty areas will look like after the chosen updates. The point where the dashed horizontal and vertical lines meet in an uncertainty area, is the precise location of the corresponding point in P. In other words this is the precise location when an area becomes trivial, revealed when both coordinates of that area have been updated.

Configuration 0. This configuration provides updates as seen in Figure 7.7. It is easy to see that if one update of U has not been made either area B remains dependent among $\mathcal{A}$ (if A^x or C^y is left out) or it changes status to partly among $\mathcal{A}$ (if B^x or B^y is left out). As by Remark 4.5 the problem is solved only if all areas are either maximal or dominated among $\mathcal{A}$, and therefore a further update is needed. As such **we use Configuration 0 to provide the first three updates requested**. While not all the updates in this configuration are given to an update strategy, this shows that any deterministic online algorithm would require a fourth update to solve the problem.

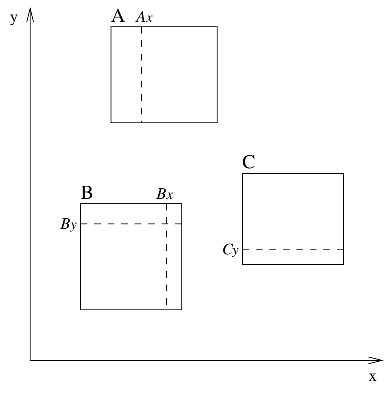


Figure 7.7: Configuration 0

We show that 4 updates are made where only one is needed as follows. For the first three update requests the strategy will return the result of the update as given in Configuration 0, regardless of the order in which the updates were requested. By our previous observation a fourth update is needed. The following 4 configurations $(1-4)$ of Figures 7.8, 7.9, 7.10 and 7.11 correspond to the situation where one update of $\{A^x, B^x, B^y, C^y\}$ was not requested in the first three updates. We now argue that the fourth update is enough by its own to determine about B.

Configuration 1. C^y is the fourth update requested. See Figure 7.8. The update of C^y reveals that the y coordinate of every point in C is greater than the y coordinate of every point in B and also every point in C already had before the updates a greater x coordinate than the x coordinate of every point in B. As such, after only updating C^y, area B changes status to dominated among $\mathcal{A}$.

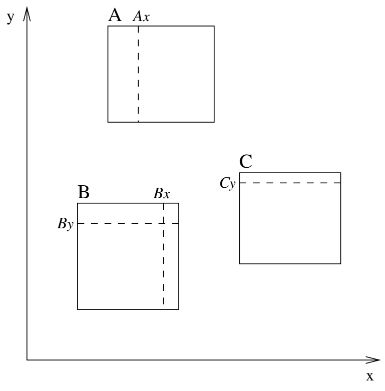


Figure 7.8: Configuration 1

Configuration 2. A^x is the fourth update requested. See Figure 7.9. The update of A^x reveals that the x coordinate of every point in A is greater than the x coordinate of every point in B and also every point in A already had before the updates a greater y coordinate than the y coordinate of every point in B. As such, after only updating A^x, area B changes status to dominated among $\mathcal{A}$.

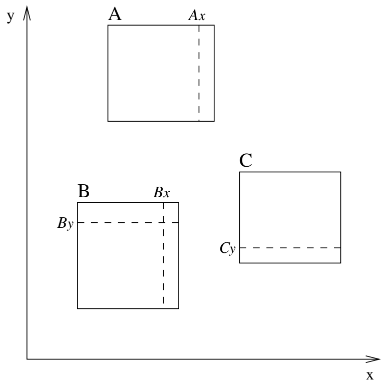


Figure 7.9: Configuration 2

Configuration 3. B^y is the fourth update requested. See Figure 7.10. The update of B^y reveals that the y coordinate of every point in B is lower than the y coordinate of every point in C and also every point in C already had before the updates a greater x coordinate than the x coordinate of every point in B. As such, after only updating B^y, area B changes status to dominated among $\mathcal{A}$.

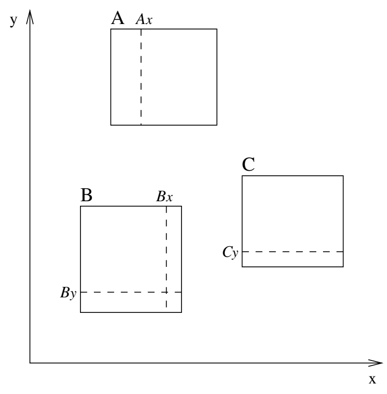


Figure 7.10: Configuration 3

Configuration 4. B^x is the fourth update requested. See Figure 7.11. The update of B^x reveals that the x coordinate of every point in B is lower than the x coordinate of every point in A and also every point in A already had before the updates a greater y coordinate than the y coordinate of every point in B. As such, after only updating B^x, area B changes status to dominated among $\mathcal{A}$.

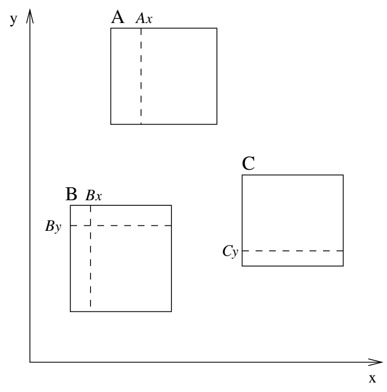


Figure 7.11: Configuration 4

For each case we have shown that the fourth update request made out of the set $U = \{A^x, B^x, B^y, C^y\}$ would have been enough to change the status of B to a dominated area among $\mathcal{A}$, even without the previously made requests. Hence, in all cases OPT is 1 and any deterministic online algorithm needs 4 updates.

To show the lower bound we simply construct multiple gadget where each is placed below and to the right of the previous one (see Figure 7.12). This guarantees that no area from one gadget contains a point higher than a point from an area of some other gadget. Hence, for each gadget, OPT performs 1 update and any deterministic online algorithm performs 4 updates. Therefore for j gadgets of this construction $OPT = j$ and any algorithm is $4j$. This concludes our proof that there does not exist a k-update competitive algorithm for the MPCS problem with $k < 4$.

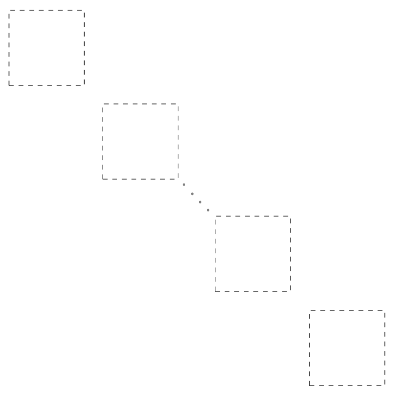


Figure 7.12: Placement of gadgets

□

7.2.3 Conclusion

Following Remark 7.6 and Lemma 7.7, we summarize our results for this section with Theorem 7.8:

Theorem 7.8. *The witness algorithm for the online problem of MPCS is 4-update competitive. Furthermore, this is the best possible.*

7.3 Verification Problem

7.3.1 Introduction

We now have a look at the problem of Maximal Point Coordinate Specific Verification (MPCSV). We still consider the model of Definition 6.1 where the areas of uncertainty are the direct product of x and y coordinate values. It is unclear at the moment if our polynomial time algorithm from Section 6.5 in Chapter 6 that solves the MPDPV problem can be modified to produce an algorithm for the MPCSV problem. We focus on the difficulties why this might not be possible. We show various figures with instances of areas where the problem becomes more complicated for MPCSV than MPDPV. In the figures we denote an uncertainty area with an upper case letter and its corresponding precise location with the matching lower case letter. The dashed lines show how an area is updated for the specified x or y coordinate.

In Section 6.5 we have combined the status of areas among $\mathcal{A}$ with the status of their precise location among P to create the new classifications of M-M/D-M/D-N/P-M/P-N/N-N. We note that these classifications also apply to the MPCSV problem, as the way the updates are performed does not affect them. Recall the classifications are as follows. Let A be an uncertainty area with precise location p, then A is:

- **M-M** if A is maximal among $\mathcal{A}$ and p is maximal among P
- **D-M** if A is dependent among $\mathcal{A}$ and p is maximal among P
- **D-N** if A is dependent among $\mathcal{A}$ and p is non-maximal among P
- **P-M** if A is partly among $\mathcal{A}$ and p is maximal among P
- **P-N** if A is partly among $\mathcal{A}$ and p is non-maximal among P
- **N-N** if A is dominated among $\mathcal{A}$ and p is non-maximal among P

7.3.2 Overview

After all areas have been classified to the above, the first action for MPDPV was to update each area $A \in \mathcal{A}$ such that $\mathcal{A} \setminus A$ is not an update solution. This was done following the reasoning of Lemma 6.14 that A must be inside every update solution, and with Lemma 6.16 we have shown that after all such areas are updated there are no D-M or P-N areas found in $\mathcal{A}$. Performing a similar action for MPCSV the equivalent would be as follows. For area A let U_{A^x} be the set that contains every update for $\mathcal{A}$ excluding A^x and similarly let U_{A^y} be the set that contains every update for $\mathcal{A}$ excluding A^y. If U_{A^x} is not an update solution (i.e. there exists an area that is neither maximal nor dominated among $\mathcal{A}$ by Remark 4.5), then every update solution must contain A^x. Similarly, if U_{A^y} is not an update solution then every update solution must contain A^y. After this action takes place we show with various examples that D-M and P-N areas can exist for MPCSV (Example 1, 2 and 3), and we further show some types of D-N areas that can exist for MPCSV but not MPDPV (Example 4 and 5), all of which create more choices in order to solve the MPCSV problem. Also with Example 6 we show for Step 2 of Algorithm 3 where one update is made for MPDPV, there can be a choice between two updates for MPCSV.

7.3.3 D-M and P-N Areas

Example 1. In Figure 7.13 area A is P-N, area B is D-M and areas C and D are both M-M. For MPDPV $\mathcal{A} \setminus A$ is not an update solution and therefore area A would be updated by Step 1 of Algorithm 3 and the problem would be solved. Whereas for MPCSV sets U_{A^x} and U_{A^y} are both update solutions and there is the choice here of either updating A^x or A^y to solve the problem. Therefore, Lemma 6.16 does not apply here as there can exist an area that is D-M or P-N after the procedure of performing all updates that have to be inside every update solution.

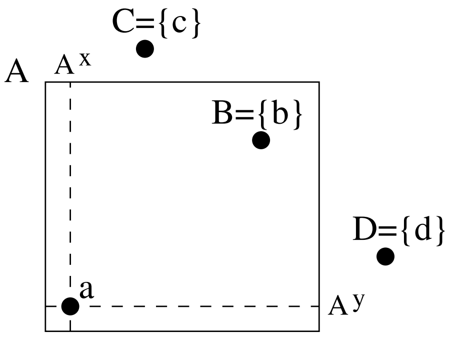


Figure 7.13: Example 1

Example 2. In Figure 7.14 area A is P-N, areas B and C are P-M and area D is D-M. For MPDPV the set $\mathcal{A} \setminus A$ is not an update solution and therefore A would be updated by Step 1 of Algorithm 3. As such the problem would be solved as areas B, C and D would change status to M-M and area A would change to N-N. For MPCSV the sets U_{A^x} and U_{A^y} are both update solutions and so there is no update that exists is every update solution, but we note the optimal update solution for this instance is $\{A^x, A^y\}$ which is consistent with MPDPV. However the sets $\{B^x, B^y, A^x\}$ and $\{C^x, C^y, A^y\}$ are also update solutions and although not optimal for this instance they could be preferable for constructing an optimal update solution if Example 2a is part of a bigger configuration of the problem. Looking at Figure 7.15, the problem is similar except updates also need to be made to change the status of the additional area E to M-M. Considering separately the problem that consists of only areas E and B, updates need to be made such that no longer there exists a point in one area higher than a point in the other. To accomplish this at least two updates need to be made with the following optimal update solutions: $\{E^x, E^y\}$, $\{B^x, B^y\}$, $\{E^x, B^y\}$ and $\{B^x, E^y\}$. Combining our observations for optimal update solutions for Example 2a and areas E and B, to solve the problem of Example 2b, the only optimal update solution is $\{B^x, B^y, A^x\}$. With a similar reasoning for Example 2c, in Figure 7.16, the optimal update solution is $\{C^x, C^y, A^y\}$. So our argument is that to determine about area A in a MPCSV problem where there exist areas such as B, C and D, a choice has to be made between the update sets $\{A^x, A^y\}$, $\{B^x, B^y, A^x\}$ and $\{C^x, C^y, A^y\}$.

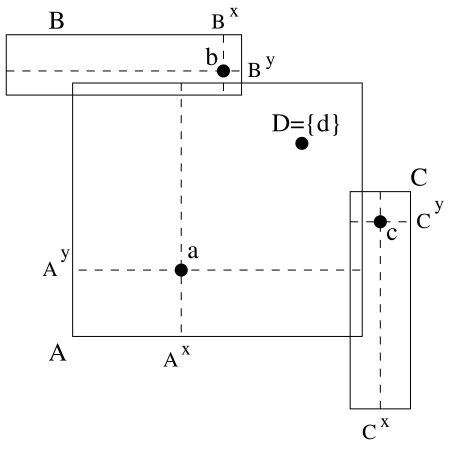


Figure 7.14: Example 2a

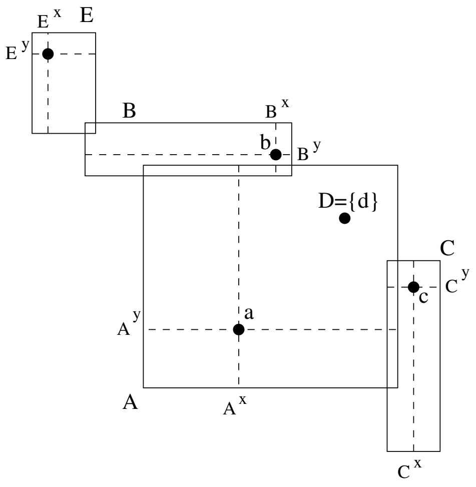


Figure 7.15: Example 2b

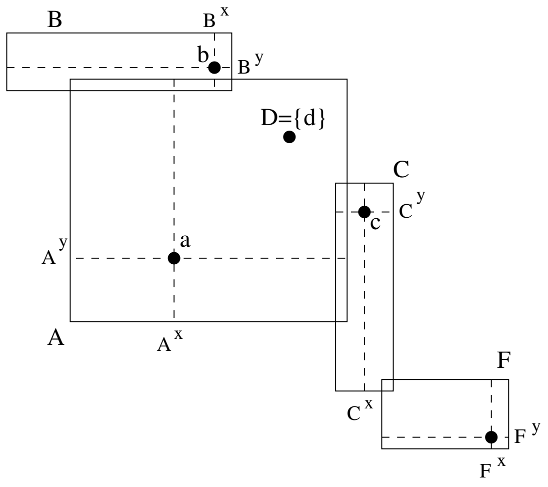


Figure 7.16: Example 2c

Example 3. In Figure 7.17 area A is P-N, areas B and C are M-M, areas D and E are P-M and area F is D-M. For MPDPV the set $\mathcal{A} \setminus A$ is not an update solution and therefore A would be updated by Step 1 of Algorithm 3. As such the problem would be solved as areas D, E and F would change status to M-M and area A would change to N-N. For MPCSV the sets U_{A^x} and U_{A^y} are both update solutions and we note there are multiple optimal update solutions: $\{A^x, A^y\}$, $\{A^x, B^x\}$, $\{A^x, D^y\}$, $\{A^y, E^x\}$ and $\{A^y, C^y\}$.

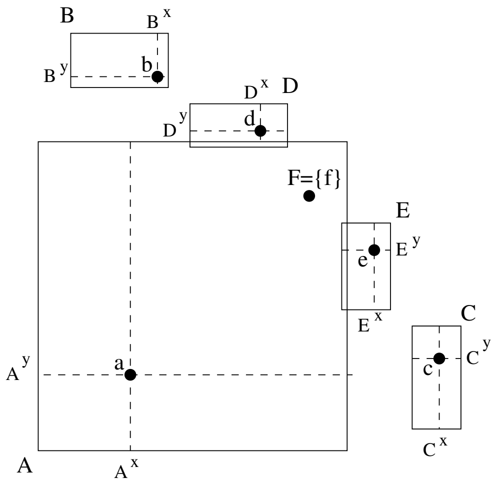


Figure 7.17: Example 3

7.3.4 D-N Areas

Example 4. In Figure 7.18 area A is D-N and areas B and C are M-M. For MPDPV the set $\mathcal{A} \setminus A$ is not an update solution and therefore A would be updated by Step 1 of Algorithm 3 and the problem would be solved. Whereas for MPCSV sets U_{A^x} and U_{A^y} are both update solutions and there is the choice here of either updating A^x or A^y to solve the problem.

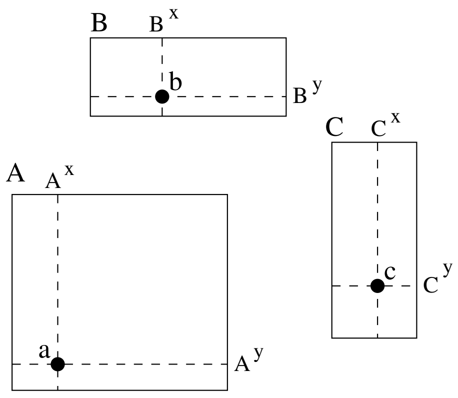


Figure 7.18: Example 4

Example 5. In Figure 7.19 area A is D-N and area B is P-M. For MPDPV the set $\mathcal{A} \setminus A$ is not an update solution (because A becomes P-N) and therefore A would be updated by Step 1 of Algorithm 3 and the problem would be solved. Whereas for MPCSV sets U_{A^x} and U_{A^y} are both update solutions and we note there are multiple optimal update solutions: $\{A^x, A^y\}$, $\{A^x, B^y\}$ and $\{A^y, B^x\}$.

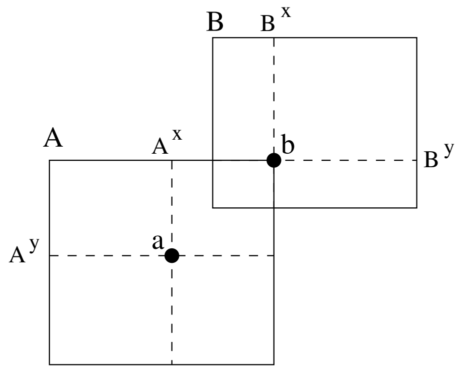


Figure 7.19: Example 5

Finally we show in Example 6 that for Step 2 of Algorithm 3 where one update is made for MPDPV, there is a choice between two updates for MPCSV.

Example 6. In Figure 7.20 area A is D-N and area B is M-M. For MPDPV the algorithm would update in this instance area B, using the reasoning of Lemma 6.22 that every update solution for an instance of the problem with these properties can be modified to include B without increasing the size. However for MPCSV there is the choice here of doing this by either updating B^x or B^y. In case areas A and B were part of a larger configuration of the problem, it is not clear by looking at only A and B if B^x or B^y is a better update, and therefore no update should be made but instead a choice recorded between $\{B^x\}$ and $\{B^y\}$.

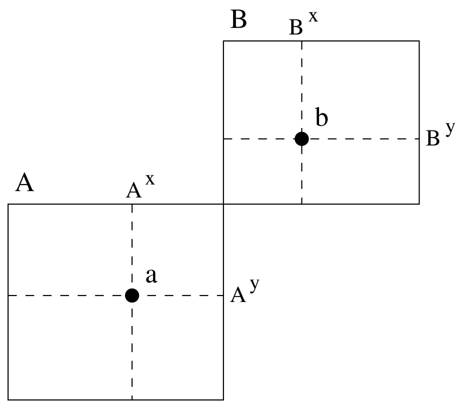


Figure 7.20: Example 6

7.3.5 Properties of Choices

Following our examples we also note the different kinds of choices encountered for MPCSV that are not present in MPDPV (recall Remark 6.34). By Example 1, 2, 3, 4 and 5 there exists in a choice a set which contains an update in an area of which the precise location is non-maximal among P. By Example 2, 3 and 5 there exist choices between 3, 4 or even 5 sets of updates. By Example 2 there exists in a choice a set of updates with size 3. By Example 2 and 3 two areas involved in different sets of updates of the same shared choice are not neighbouring (recall Definition 6.18 of neighbouring areas).

7.3.6 Conclusion

For MPCSV we have shown that D-M, P-N and some types of D-N areas can exist after performing all updates that must be inside every update solution, which is not the case for MPDPV. Furthermore for MPDPV, after Algorithm 3 finishes execution, we highlighted in Remark 6.34 the properties of the choices collected. Phase 2, in Subsection 6.5.3, provides a solution for the MPDPV problem based on those properties and as they are not consistent with the properties shown in Subsection 7.3.5, our solution for the MPDPV problem cannot be trivially converted to produce a solution for the MPCSV problem.

Chapter 8

Maximal Points - Higher Dimensions

So far we have studied the maximal points problem under various settings, however all settings involved 2-dimensional points. We now have a look at the problem where more than 2 dimensions are considered, specifically we study the maximal point under uncertainty problem in 3-dimensional space. We will give a construction for the online problem showing that a constant update competitive ratio is impossible.

We will use our notation as in Chapter 4 which naturally extends to higher dimensions, with the difference being a point $p = (p_x, p_y, p_z)$ is *higher* than a point $q = (q_x, q_y, q_z)$ if $p_x \geq q_x$ and $p_y \geq q_y$ and $p_z \geq q_z$ and $p \neq q$. Also recall set $\mathcal{A}$ contains all the areas of uncertainty and set P contains the precise location for each area in $\mathcal{A}$.

Constructing a single gadget. We first construct a gadget with n uncertainty areas. Let some $a \geq 1$, and let some b be sufficiently large such that $b \geq n$. We place uncertainty areas $A_1, A_2, A_3, \ldots, A_n$ where for each we fix the coordinates x and y to be precise and the coordinate z to be uncertain as follows: $A_1 = (a + 1, b - 1, [1, 3])$, $A_2 = (a + 2, b - 2, [1, 3])$, $A_3 = (a + 3, b - 3, [1, 3])$, $\ldots$, $A_n = (a + n, b - n, [1, 3])$. For convenience we group these areas together in the set $A = \{A_1, A_2, A_3, \ldots, A_n\}$. We further construct a trivial area B such that $B = (a, b - n - \epsilon, 2)$, with some $0 < \epsilon < 1$. So the construction so far contains $i + 1$ areas and we highlight the following properties.

The areas in A are placed in the construction in such a way that for each area in A, any other area in $\mathcal{A}$ that has a greater x coordinate also has a lower y coordinate, and each that has a greater y coordinate also has a lower x coordinate. Furthermore there are no two areas in A that have same x or the same y coordinate and these coordinates are precise. Therefore each area in A consists only of points such that there does not exist a point in some area of $\mathcal{A}$ that is higher, and we note that this is the case regardless of what their coordinate z is set to be. As such we note the following remark:

Remark 8.1. *Every area in set A is maximal among $\mathcal{A}$.*

Let some area $A_k \in A$. As by the construction, we have that A_k has both x and y coordinates precise and greater than those of area B. Also area A_k has an uncertain coordinate z that contains a value equal or greater than the z coordinate of area B. Therefore there exists a configuration where the precise location of A_k is higher than the precise location of B. In other words:

Remark 8.2. *Every area in set A contains a point higher than area B.*

However the coordinate z of area A_k also contains a value lower than the coordinate z of area B. Therefore there exists a configuration where area B is maximal among $\mathcal{A}$, this is potentially the case where every area in A is updated with the z coordinate being reduced to a value less than 2. As such updates are needed to solve the problem constructed as:

Remark 8.3. *Area B is neither maximal nor dominated among $\mathcal{A}$.*

We can assume that the precise location $p \in B$, and recall B is trivial, is non-maximal among P. However, as there exist configurations such that area B is maximal among $\mathcal{A}$, updates are needed to show that there exists an area that is higher than B, or more specifically we need to show an area in A has a greater z coordinate than the z coordinate of B. Therefore, in order to solve the problem constructed, any online strategy has to update at least one of the areas in A. For the first $n-1$ update requests, the adversary reveals that the z coordinate of all the updated areas is lower than the z coordinate of area B, and therefore a further update is needed. The nth update request reveals that an area of A has a greater z coordinate than B which solves the problem showing that B is dominated among $\mathcal{A}$. However this last update by its own would have been enough, so for our construction with input n we have that OPT performs 1 update whereas any deterministic online algorithm can be forced to perform n updates.

Constructing multiple gadgets. We now show how to replicate this construction with multiple gadgets such that they do not interfere with each other. Let m be the number of gadgets. As before each gadget contains n uncertainty areas and let some $a \geq 1$. Further let some b be sufficiently large such that $b > nm$. We construct the uncertainty areas for each $1 \leq i \leq n$ and $1 \leq j \leq m$ as $A_i^j = (a + n(j-1) + i,\ b - n(j-1) - i,\ [1,3])$ and let each group of areas be the set A^j accordingly. Finally, we also construct for each $1 \leq j \leq m$ a trivial area $B_j = (a + n(j-1) + \epsilon,\ b - jn - \epsilon,\ 2)$.

Example of a construction. An example of a construction with $m = 4$, $n = 3$, $a = 1$, $b = 13$ and $\epsilon = 0.1$ will give us the following uncertainty areas:

Gadget 1:

$A^1_1 = (2,\ 12,\ [1,3])$
$A^1_2 = (3,\ 11,\ [1,3])$
$A^1_3 = (4,\ 10,\ [1,3])$
$B_1 = (1.1,\ 9.9,\ 2)$

Gadget 2:

$A^2_1 = (5,\ 9,\ [1,3])$
$A^2_2 = (6,\ 8,\ [1,3])$
$A^2_3 = (7,\ 7,\ [1,3])$
$B_2 = (4.1,\ 6.9,\ 2)$

Gadget 3:

$A^3_1 = (8,\ 6,\ [1,3])$
$A^3_2 = (9,\ 5,\ [1,3])$
$A^3_3 = (10,\ 4,\ [1,3])$
$B_3 = (7.1,\ 3.9,\ 2)$

Gadget 4:

$A^4_1 = (11,\ 3,\ [1,3])$
$A^4_2 = (12,\ 2,\ [1,3])$
$A^4_3 = (13,\ 1,\ [1,3])$
$B_4 = (10.1,\ 0.9,\ 2)$

Result of an update. Let a gadget j be made as by our construction with the set of areas A^j and area B_j. To determine about area B_j we only consider updates made on the areas of A^j, as only these areas have x and y (and also potentially z) coordinate values greater than those of B_j, and therefore can change the status of B_j to dominated among $\mathcal{A}$. The first $n-1$ updates made on the gadget reduce the z coordinate of each area in the set A^j to be a value $2-\epsilon$ (in our example above this is 1.9). The nth update that has been made in the gadget reduces the z coordinate of an area to be a value $2+\epsilon$ (in our example above this is 2.1).

Properties of areas. We now analyse the various area properties in the construction. We begin with the following lemma:

Lemma 8.4. *Let some area $C \in A^j \cup \{B_j\}$ for some gadget j with the set of areas A^j and an area B_j as by the construction. Further let some area $D \in \mathcal{A} \setminus (A^j \cup \{B_j\})$. Then either area C has greater x coordinate and lower y coordinate than area D or area C has lower x coordinate and greater y coordinate than area D.*

Proof. This is quite easy to see. Recall that for all areas considered in the construction, there is no uncertainty about the x and y coordinates. When constructing the areas we have the pattern of increasing x and decreasing y coordinates, which follows continuously for all gadgets. □

We show that our established Remark 8.1, Remark 8.2 and Remark 8.3 also apply for $m > 1$ number of gadgets for our construction. For those three remarks we replace set A with set A^j and also replace area B with area B_j, for each $1 \leq j \leq m$. Let some area $A_i^j \in A^j$ for some i. Following the reasoning for Remark 8.1 we know that area A_i^j is maximal among $A^j \cup \{B_j\}$. As by Lemma 8.4 it also follows that **area $\mathbf{A_i^j}$ is maximal among $\mathcal{A}$**. Similarly, following the reasoning for Remark 8.2, **area $\mathbf{A_i^j}$ contains a point higher than area $\mathbf{B_j}$**. We give the following lemma to establish our last property:

Lemma 8.5. *Let j be some gadget with the set of areas A^j and an area B_j as by the construction. Then B_j is neither maximal nor dominated among $\mathcal{A}$ and updating all areas in $\mathcal{A} \setminus A^j$ does not change the status of area B_j to either maximal or dominated among $\mathcal{A}$.*

Proof. Following the reasoning for Remark 8.3, we have that area B_j is neither maximal nor dominated among $A^j \cup \{B_j\}$. Assume that area B_j is either maximal or dominated among $\mathcal{A}$. We look at each case separately:

Case B_j is maximal among $\mathcal{A}$: This implies that B_j is maximal among $A^j \cup \{B_j\}$ as well, a contradiction.

Case B_j is dominated among $\mathcal{A}$: There must exist an area $C \in \mathcal{A} \setminus (A^j \cup \{B_j\})$ such that every point in C has one coordinate greater than B_j and all other not lower than B_j. This is a contradiction as, by Lemma 8.4, either area C has a greater x coordinate and a lower y coordinate than B_j, or area C has a lower x coordinate and a greater y coordinate than B_j.

As both cases come to a contradiction, B_j is neither maximal nor dominated among $\mathcal{A}$, and we now argue that updating all areas in $\mathcal{A} \setminus A^j$ does not change the status of area B_j to either maximal or dominated among $\mathcal{A}$.

Assume that after updating all areas in $\mathcal{A} \setminus A^j$, area B_j becomes either maximal or dominated among $\mathcal{A}$. We look at each case separately:

Case B_j becomes maximal among $\mathcal{A}$: After the updates there cannot exist a point in an area of $\mathcal{A}$ that is higher than B_j. This is a contradiction as, by our previous property based on Remark 8.2, every area in A^j contained a point higher than B_j before the updates and they have not been updated.

Case B_j becomes dominated among $\mathcal{A}$: After the updates there must exist an area $C \in \mathcal{A} \setminus A^j$ such that every point in C has one coordinate greater than B_j and all other not lower than B_j. By Lemma 8.4, before the updates either area C had a greater x coordinate and a lower y coordinate than B_j, or area C had a lower x coordinate and a greater y coordinate than B_j. As all x and y coordinates of all areas were precise before the updates, it follows that these coordinates are unaffected after the updates. A contradiction.

As both cases come to a contradiction, updating all areas in $\mathcal{A} \setminus A^j$ does not change the status of area B_j to either maximal or dominated among $\mathcal{A}$.

□

With Lemma 8.5 we have effectively shown that any update in a gadget does not affect the status of an area among $\mathcal{A}$ found is some other gadget. Further we have established how to create an arbitrary number m of gadgets where for each gadget $OPT = 1$ but any deterministic online update strategy makes n updates. Hence for the whole construction $OPT = m$ where an update strategy makes mn updates. Following these properties of our construction we give our result:

Theorem 8.6. *There does not exist a deterministic online algorithm with constant competitive ratio for the 3-dimensional maximal points under uncertainty problem.*

We note that it is also trivial that our result is carried over to dimensions higher than 3D. Furthermore this is also the case for when either open or closed intervals are considered and even for areas of uncertainty that contain at most 2 points. Our result motivates the investigation of new ways to restrict the areas of uncertainty when considering the maximal point problem in higher than 2 dimensions.

Chapter 9

Farthest Pair of Vertices

9.1 Introduction

In this chapter we study graphs which are weighted, undirected and connected. With this in mind a graph contains vertices and edges such that there exists a path of edges between any two vertices and each edge has the attribute of weight. The distance then between two vertices is the sum of all edge weights found along a path between them, such that the sum is minimum among all paths between the two vertices. The distance between a pair of vertices that is the maximum among the distances of all pairs of vertices is said to be the diameter of the graph and this pair of vertices is a farthest pair of vertices. We note that there could be multiple such pairs.

Under uncertainty, the idea is that each edge weight could be given in an uncertain form and update operations can be made on edge weights which obtain the precise measurement. We are interested in finding a farthest pair of vertices with this uncertainty element in mind, which is directly related to the classical problem of finding the diameter of a graph. In our case, however, the task does not involve the actual value of the diameter, but the task is rather to perform enough updates such that a farthest pair of vertices is computable.

Motivation. This kind of graph problems rise naturally in problems such as finding in a wireless network the most expensive pair of communicating devices. Expenses could be in terms of transmitter output power needs, which is affected for example by distance or other physical obstructions. Uncertainty can be modelled here as the devices may often be in motion but guaranteed to not travel far from a specific location. Finding the most expensive pair of devices could be useful if one wants to make changes to reduce overall expenses.

Outline. In Section 9.2 we give formal definitions. In Section 9.3 we study the problem for trees, specifically we give matching upper and lower bounds in Subsection 9.3.1 under the assumption that each uncertainty area is either a singleton or an open interval and in Subsection 9.3.2 we show the lower bound if closed intervals are allowed. In Section 9.4 we show the lower bound for cycle graphs.

9.2 Preliminaries

We first give definitions for the problem of finding a farthest pair or vertices in a graph without uncertainty. We write $G = (V, E, w)$ for a *weighted graph* with vertex set V, edge set $E \subseteq V \times V$ and weight function $w : E \to \mathcal{R}$. We also write w_e for $w(e)$ of an edge $e \in E$.

The length of a path between two vertices in G is the sum of weights for all edges in the path. The *distance* between two vertices is the length of a smallest path. As we noted before, we only consider undirected graphs and further all edge weights are non-negatives. We use the following to refer to the distance between two vertices:

Definition 9.1. *For two vertices u, v of a weighted graph (V, E, w), we write $d_w(u, v)$ for the distance between u and v.*

A pair of vertices (u, v) in a weighted graph G is said to be a *farthest pair of vertices* if there is no pair of vertices such that the distance between them is greater than the distance between u and v. Formally:

Definition 9.2. *In a weighted graph (V, E, w) with vertices $u, v \in V$, the pair (u, v) is a farthest pair of vertices such that $d_w(u, v) \geq d_w(x, y)$ for every $x, y \in V$.*

We now use the under uncertainty setting in the context of weighted graphs, as by our definitions in Section 2.2. Let (S, U, A, w) be an instance of the farthest pair of vertices under uncertainty problem. Then $S = (V, E, w)$ is the weighted graph as above and U consists of the edges that have uncertain weight. For each $u \in U$ function w maps u to its precise edge weight w_u and area function A maps u to A_u. Throughout this chapter we assume that A_u is an open interval, apart from Subsection 9.3.2 where it can also be a closed intervals. Here the set $\phi(S, U, w)$ contains all the farthest pairs of vertices. For the remainder of this chapter we redefine for convenience an instance of the problem to (V, E, w, A), and also define an *edge uncertainty graph* as $\mathcal{U} = (V, E, A)$ from the above. We further say a weighted graph (V, E, w) is *consistent* with an uncertainty graph (V, E, A) if for every $e \in E$ we have that $w_e \in A_e$. This brings us to the two following definitions:

Definition 9.3. *An instance of the Farthest Pair of Vertices under uncertainty problem (FPV for short) consists of (V, E, w, A) where (V, E, w) is a weighted graph that is consistent with the uncertainty graph (V, E, A).*

Definition 9.4. *An edge uncertainty graph $\mathcal{U} = (V, E, A)$ is solved if there exists $u, v \in V$ such that (u, v) is a farthest pair of vertices in every weighted graph that is consistent with $\mathcal{U}$.*

And we further note *an instance (V, E, w, A) of FPV is solved* if the uncertainty graph (V, E, A) is solved.

In an instance (V, E, w, A) of FPV *updating* an edge $e \in E$ changes the instance to (V, E, w, A') where $A'_u = A_u$ for every $e \neq u$ and $A'_e = \{w_e\}$. We also say w_e is the *actual weight* of the edge e. The input of an algorithm is (V, E, A) (we only study the online setting in this chapter) and hence the aim is to keep updating edges iteratively, until the instance of the problem becomes a solved instance or in other words until *a farthest pair of vertices is computable* on (V, E, A).

Any set of updates that changes the initial instance of the problem to a solved instance is called an update solution. An update solution of minimal size is also called an optimal update solution and we denote the number of elements in this case by OPT. An algorithm for the FPV problem is evaluated by comparing the size of the update solution it produces in the worst case against OPT. This is either expressed as a ratio or by a bounding function over OPT.

9.3 FPV for Trees

Introduction. As stated in Section 9.1 we consider a specific type of graphs for the FPV problem such that they are weighted, undirected and connected. In addition to that, in this section we look specifically at trees. Naturally, here, any farthest pair of vertices are leaves, under the assumption that all edge weights are non-negative. We use the abbreviation FPVT for the Farthest Pair of Vertices in a Tree under uncertainty, which is in line with Definition 9.3. We look at two different restrictions for the uncertainty areas of the FPVT problem. In Subsection 9.3.1 we assume that each uncertainty area is either trivial or an open interval, and in Subsection 9.3.2 we assume that this is not the case.

We use throughout this section the following notation:

Definition 9.5. *For two vertices u, v in a tree, we write $P(u, v)$ for the set of all edges that lie on the path between u and v.*

Definition 9.6. *In an edge uncertainty graph $\mathcal{U} = (V, E, A)$ we say the distance between two vertices $u, v \in V$ is potentially greater than the distance between two vertices $x, y \in V$, if there exists a weighted graph $G = (V, E, w)$ consistent with $\mathcal{U}$ such that $d_w(u, v) > d_w(x, y)$. We write for convenience $(u, v) >^? (x, y)$. Similarly we write $(u, v) \geq^? (x, y)$ for $d_w(u, v) \geq d_w(x, y)$.*

Definition 9.7. *Let $\mathcal{U} = (V, E, A)$ be an edge uncertainty tree with some $u, v \in V$ and for any $e \in E$ let L_e denote the lower limit and U_e the upper limit of the uncertainty area A_e. Further let $S_{u,v}^{min} = \sum_{e \in P(u,v)} L_e$ and $S_{u,v}^{max} = \sum_{e \in P(u,v)} U_e$. We write $d_A(u, v) = (S_{u,v}^{min}, S_{u,v}^{max})$ if $P(u, v)$ contains an edge with uncertainty area which is an open interval and otherwise we write $d_A(u, v) = [S^{min}, S^{max}]$. We also say this is the uncertainty distance between u and v.*

Note on Definition 9.7: We use this definition to obtain the lower and upper limit of where the actual distance between two vertices lies. This can be used for any combination of types of uncertainty areas for consecutive edges in a set; whether they are open intervals, closed intervals or trivial (a trivial area is basically the same as a closed interval which has the same lower and upper limit).
For vertices x, y, u, v in an edge uncertainty graph, we write $d_A(x, y) \geq d_A(u, v)$ if the sum of the lowest possible values of all uncertainty areas of edges in $P(x, y)$, is at least as great as the sum of the highest possible values of all uncertainty areas of edges in $P(u, v)$. Furthermore we use this definition throughout this section to make comparisons of uncertainty distances only where $P(x, y) \cap P(u, v) = \emptyset$.
Specifically we compare uncertainty distances in the following way. Let $S_{x,y}^{min}$ be the lower limit of $d_A(x, y)$ and let $S_{u,v}^{max}$ be the upper limit of $d_A(u, v)$. If $S_{x,y}^{min} > S_{u,v}^{max}$ then obviously $d_A(x, y) > d_A(u, v)$. For $S_{x,y}^{min} = S_{u,v}^{max}$ we have the following 3 cases: If at least one edge in $P(x, y) \cup P(u, v)$ has an open interval for an uncertainty area then $d_A(x, y) > d_A(u, v)$. If all edges in $P(x, y) \cup P(u, v)$ have trivial uncertainty areas then $d_A(x, y) = d_A(u, v)$. Otherwise there must exist a non-trivial closed interval uncertainty area and then $d_A(x, y) \geq d_A(u, v)$.
We may also write $d_A(x, y) \ngeq d_A(u, v)$ or $d_A(x, y) \ngtr d_A(u, v)$ to show that the relations cannot hold.

Definition 9.8. *The hopping diameter in a tree is the maximal number of edges that any two vertices are apart.*

The following is a general property of trees:

Lemma 9.9. *A tree with hopping diameter k and l leaves has at most $\left\lfloor \frac{k}{2} \right\rfloor \cdot l + (k \mod 2)$ edges.*

Proof. **Base Case:** For $l = 1$ leaf there are no edges in the tree and the hopping diameter is $k = 0$. So, as $0 \leq \left\lfloor \frac{k}{1} \right\rfloor \cdot 2 + (k \mod 2)$, lemma is correct for $l = 1$. For $l = 2$ leaves the tree is a line for any hopping diameter k and the total number of edges must be the same as the hopping diameter. So as $k \leq \left\lfloor \frac{k}{2} \right\rfloor \cdot 2 + (k \mod 2)$ lemma is correct for $l = 2$ and any k.

Induction Hypothesis: Assume lemma is correct for $l \geq 2$ leaves.

Inductive step: Let $G = (V, E)$ be a tree with hopping diameter k and $l + 1$ leaves.

We need to show $|E| \leq \left\lfloor \frac{k}{2} \right\rfloor \cdot (l + 1) + (k \mod 2)$.

We note that since G contains at least 3 leaves then there exists a vertex with degree at least 3. Among all pairs of vertices in G where one vertex is a leaf and the other vertex has degree at least 3, let (x, y) be a pair of vertices such that $|P(x, y)|$ is the smallest. Without loss of generality let x be the leaf and y have degree 3 or more.

We argue that $|P(x, y)| \leq \left\lfloor \frac{k}{2} \right\rfloor$: Assume that $|P(x, y)| > \left\lfloor \frac{k}{2} \right\rfloor$. Since $|P(x, y)|$ is minimal, for each leaf z in G then $|P(z, y)| \geq |P(x, y)| > \left\lfloor \frac{k}{2} \right\rfloor$. As there is a unique path between any two vertices then $|P(x, z)| = |P(x, y)| + |P(z, y)| \geq \left\lfloor \frac{k}{2} \right\rfloor + 1 + \left\lfloor \frac{k}{2} \right\rfloor + 1 \geq k + 1$. A contradiction as the hopping diameter of G is k, therefore $|P(x, y)| \leq \left\lfloor \frac{k}{2} \right\rfloor$.

Now let $G' = (V', E')$ be the tree after removing from G all edges of $P(x, y)$ along with all vertices in their path except vertex y. So we have taken away leaf x along with at most $\left\lfloor \frac{k}{2} \right\rfloor$ edges. See Figure 9.1.

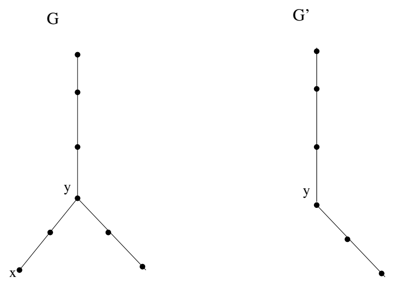


Figure 9.1: Tree G' from G

Let k' be the hopping diameter of G'.

By the Induction Hypothesis $|E'| \leq \left\lfloor \frac{k'}{2} \right\rfloor \cdot l + (k' \mod 2)$ for every $l \geq 2$

We argue that $\left\lfloor \frac{k'}{2} \right\rfloor \cdot l + (k' \mod 2) \leq \left\lfloor \frac{k}{2} \right\rfloor \cdot l + (k \mod 2)$: We know that $k' \leq k$ as we have only removed some edges and vertices to a tree and have not added anything. Let A be the left-hand side and B be the right-hand side of the inequality. If $k' = k$ then $A = B$ and so it remains to show for $k' < k$. We separate for $k' < k$ the cases where 1) k' is odd and k is odd, 2) k' is odd and k is even, 3) k' is even and k is odd, 4) k' is even and k is even. So B is greater than A by at least 1) $l + 0$, 2) $l - 1$, 3) $0 + 1$ and 4) $l + 0$. As $l \geq 2$ the inequality holds for every case.

Then $|E| \leq |E'| + \left\lfloor \frac{k}{2} \right\rfloor \leq \left\lfloor \frac{k'}{2} \right\rfloor \cdot l + (k' \mod 2) + \left\lfloor \frac{k}{2} \right\rfloor \leq \left\lfloor \frac{k}{2} \right\rfloor \cdot l + (k \mod 2) + \left\lfloor \frac{k}{2} \right\rfloor = \left\lfloor \frac{k}{2} \right\rfloor \cdot (l + 1) + (k \mod 2)$

And so the lemma holds for any l leaves and hopping diameter k.

□

9.3.1 Open Intervals

9.3.1.1 Upper Bound

We propose an online update strategy for solving the FPVT problem under the restriction of trivial or open uncertainty areas, using a similar technique as by Erlebach et al. [17]. We will compare the number of updates performed by the algorithm against OPT, showing that the algorithm builds an update solution of size at most $\lfloor \frac{k}{2} \rfloor \cdot OPT + (k \mod 2) + k$ where k is the hopping diameter of the graph.

Before giving the algorithm we establish some notations and properties that are needed to define and analyse the algorithm.

First we establish the following property of FPVT:

Lemma 9.10. *Let $\mathcal{U} = (V, E, A)$ be a solved edge uncertainty graph and let $G = (V, E, w)$ be a weighted graph that is consistent with $\mathcal{U}$. If (x, y) is a farthest pair of vertices in G then (x, y) is a farthest pair of vertices in every weighted graph that is consistent with $\mathcal{U}$.*

Proof. As $\mathcal{U}$ is a solved edge uncertainty graph there exists a pair of vertices (u, v) that is a farthest pair of vertices in every weighted graph that is consistent with $\mathcal{U}$.

Let (x, y) be some other pair of vertices that is a farthest pair of vertices in G.

Note that as we are only considering trees there is a unique path between any two vertices and we use $P(u, v)$ for the set of edges between u, v and similarly $P(x, y)$ for the set of edges between x, y. We now distinguish the following two cases for edge sets $P(u, v)$ and $P(x, y)$:

Case 1: Assume there exists an edge $f \in P(u, v) \setminus P(x, y)$ with non trivial area of uncertainty. Then let w' be a new weight function which is identical to w except for the edge f such that $w'_f < w_f$ with $w'_f \in A_f$. Such a function exists as A_f is non trivial and hence open. The weighted graph (V, E, w') is also consistent with $\mathcal{U}$ and as $d_{w'}(u, v) < d_{w'}(x, y)$ the pair (u, v) is not a farthest pair of vertices in (V, E, w'). A contradiction.

Case 2: Assume there exists an edge $f \in P(x, y) \setminus P(u, v)$ with non trivial area of uncertainty. Then let w'' be a new weight function which is identical to w except for the edge f such that $w''_f > w_f$ with $w''_f \in A_f$. Such a function exists as A_f is non trivial and hence open. The weighted graph (V, E, w'') is also consistent with $\mathcal{U}$ and as $d_{w''}(u, v) < d_{w''}(x, y)$ the pair (u, v) is not a farthest pair of vertices in (V, E, w''). A contradiction.

By combining Case 1 and Case 2, all edges in $P(x, y) \triangle P(u, v)$ are trivial and the distance between u, v is the same as the distance between x, y in all weighted graphs that are consistent with $\mathcal{U}$. Hence (x, y) must also be a farthest pair of vertices in every weighted graph that is consistent with $\mathcal{U}$.

□

Following Lemma 9.10, Definition 9.4 and Definition 9.6 we remark:

Remark 9.11. *Let $I = (V, E, w, A)$ be an instance of the FPVT problem such that (u, v) is a farthest pair of vertices in (V, E, w). If there exist two vertices x, y such that $(x, y) >^? (u, v)$ then I is not solved.*

We define the following to be used by the algorithm for updates:

Definition 9.12. *Let $\mathcal{U} = (V, E, A)$ be an edge uncertainty graph where (u, v) is a pair of vertices. We say (u, v) is a Robust Potential Farthest Pair (RPFP) if for all possible weights of edges in $E \setminus P(u, v)$ there exist possible weights for the edges in $P(u, v)$ such that (u, v) is a farthest pair of vertices in the resultant weighted graph.*

And we use the following for to obtain a set of edges in a tree:

Definition 9.13. *For two pairs of vertices (x, y) and (u, v) in a tree, we define Q_x and Q_y to be the edge sets as follows:*

$$Q_x = P(x, y) \setminus (P(u, y) \cup P(v, y))$$

$$Q_y = P(x, y) \setminus (P(u, x) \cup P(v, x))$$

And we note $Q_x \cap Q_y = \emptyset$.

Furthermore for an edge uncertainty graph we define $\bar{Q}_x$ to be the subset of Q_x such that $\bar{Q}_x$ only contains edges with non-trivial uncertainty area.

At this point we recall the concept of a witness set for problems under uncertainty which we will use for the FPVT problem: a witness set is a set of uncertainty areas such that at least one element from the set needs to be inside every update solution.

We argue why the following is a witness set and then we give our algorithm:

Lemma 9.14. *Let (x, y) be a RPFP in an edge uncertainty graph $\mathcal{U} = (V, E, A)$, and let $I = (V, E, w, A)$ be an instance of the FPVT problem such that (u, v) is a farthest pair of vertices in (V, E, w) with $x \notin \{u, v\}$. Further let Q_x be the edge set for the two pairs (x, y) and (u, v). Then $\bar{Q}_x$ is either empty or a witness set for I.*

Proof. The set Q_x either contains an edge with non-trivial uncertainty area in $\mathcal{U}$ (an open interval by our restriction) or it consists of only edges that have trivial uncertainty area and thus $\bar{Q}_x$ is empty. So it remains to show that if $\bar{Q}_x$ is not empty then it is a witness set, and therefore for the rest of the proof lets assume Q_x contains at least one edge with non-trivial uncertainty area. Moreover let $\mathcal{U}' = (V, E, A')$ be the edge uncertainty graph after updating in $\mathcal{U}$ all edges of $E \setminus P(x, y)$. We will use the potentially greater notation of Definition 9.6 for (x, y) and (u, v) in $\mathcal{U}'$.

We recall that, as we are only considering trees, there is a unique path between any two vertices and separate the cases of $y \in \{u, v\}$ in Case 1 and $y \notin \{u, v\}$ in Case 2.

Case 1: $y \in \{u, v\}$**.** Without loss of generality lets assume $y = v$ and let a be the vertex that joins the two paths, see example in Figure 9.2.

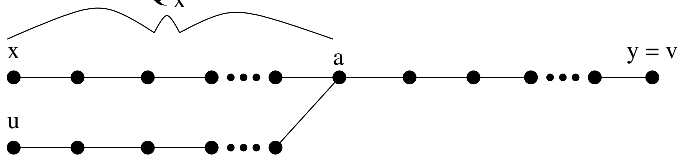


Figure 9.2: Case 1

Since (x, y) is a RPFP then we have $(x, y) \geq^? (u, v)$. By our assumption Q_x contains an edge e such that A_e is an open interval, and we have $e \in P(x, y) \setminus P(u, v)$. Therefore, similarly with the arguments in the proof of Lemma 9.10, we have $(x, v) >^? (u, v)$. That is, as A_e is an open interval, there exists a weighted graph (V, E, w'') consistent with $\mathcal{U}'$ such that $d_{w''}(x, v) > d_{w''}(u, v)$.

It must hold that $(x, v) >^? (u, v)$ even after further updating all edges in $P(a, y)$ since the sub-path between a, y is shared among the paths x, y and u, v and hence, by Remark 9.11, $\bar{Q_x}$ is a witness set.

Case 2: $y \notin \{u, v\}$**.** By our assumption $x \notin \{u, v\}$ and so we note the path between x, y does not share endpoints with the path between u, v. In Figure 9.3 we can see an example where the two paths do not share edges (Case 2a) and in Figure 9.4 another where they share an edge (Case 2b). Let a be the vertex such that $P(x, a) = Q_x$. For Case 2a let b be the vertex such that $P(a, b) = P(x, u) \setminus (P(x, y) \cup P(u, v))$ and for Case 2b let $a = b$, to help our analysis apply for both.

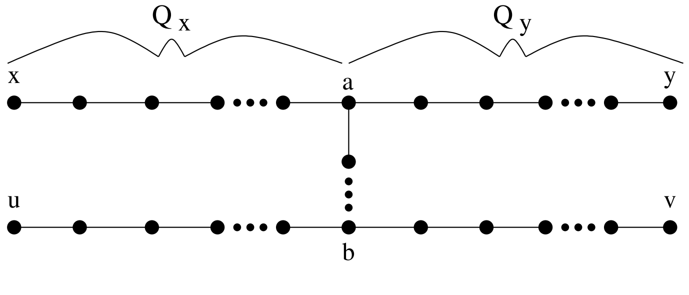


Figure 9.3: Case 2a

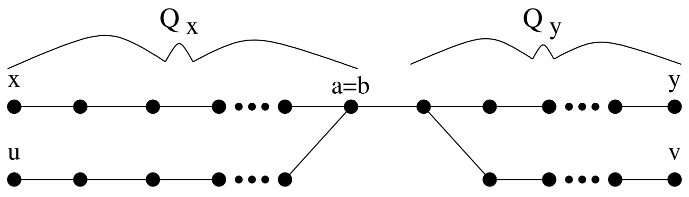


Figure 9.4: Case 2b

We give the following analysis which covers both Case 2a and Case 2b. Since (x, y) is a RPFP then we have $(x, y) \geq^? (u, y)$. As the sub-path between a, y is shared among the paths between x, y and u, y, then also $(x, a) \geq^? (u, a)$. It follows that $(x, b) \geq^? (u, b)$ by switching the sub-path between a, b to be contained in the left-hand side of the inequality instead of the right for Case 2a, and for Case 2b we had $a = b$. By including the sub-path between b, v for both sides we have $(x, v) \geq^? (u, v)$. By our assumption Q_x contains an edge e such that A_e is an open interval, and we have $e \in P(x, v) \setminus P(u, v)$. Therefore, similarly with the arguments in the proof of Lemma 9.10, we have $(x, v) >^? (u, v)$. That is, as A_e is an open interval, there exists a weighted graph (V, E, w'') consistent with $\mathcal{U}'$ such that $d_{w''}(x, v) > d_{w''}(u, v)$.

It must hold that $(x, v) >^? (u, v)$ even after further updating all edges in $P(a, y)$ as this set only contains edges that are either inside $P(x, v) \cap P(u, v)$ or not inside $P(x, v) \cup P(u, v)$ and hence, by Remark 9.11, $\bar{Q_x}$ is a witness set.

□

Algorithm action. A RPFP can be found for every edge uncertainty graph. We outline our algorithm as follows:

Algorithm 6 The witness algorithm for FPVT

```
1: while Current edge uncertainty graph is not solved do
2:     Find a RPFP $(x, y)$;
3:     Update every edge in the set $P(x, y)$.
4: end while
```

Algorithm analysis. For our analysis we do not argue about the maximum size of the set of updates the algorithm makes in any iteration, as usually done in uncertainty problems, but rather compare against OPT the overall number of updates the algorithm made after it has finished execution. This brings us to the following lemma:

Lemma 9.15. *Under the restriction of open or trivial uncertainty areas and k hopping diameter, Algorithm 6 makes at most $\left\lfloor \frac{k}{2} \right\rfloor \cdot OPT + k + (k \mod 2)$ updates for any instance the FPVT problem.*

Proof. Let (V, E, w, A) be an instance of the problem and let (u, v) be a farthest pair of vertices in (V, E, w). In each iteration the algorithm finds a RPFP (x, y) for (V, E, A) and updates every edge in $P(x, y)$.

By Definition 9.13, for pairs (x, y) and (u, v), we have the edge sets Q_x and Q_y and their respective subsets $\bar{Q_x}$ and $\bar{Q_y}$. By Lemma 9.14, for $x \notin \{u, v\}$ we have that $\bar{Q_x}$ is either empty or a witness set and similarly for $y \notin \{u, v\}$ we have that $\bar{Q_y}$ is either empty or a witness set.

Let l be the number of different vertices encountered in RPFPs in the full run of the algorithm, such that for pairs (x, y) and (u, v) the set $\bar{Q_x}$ was not empty (i.e. at least one edge of Q_x had a non-trivial uncertainty area). So at least $l - 2$ such vertices are not u or v.

Updates by OPT. Therefore there have been at least $l-2$ witness sets updated and, by Remark 2.1, $OPT \geq l - 2$.

Updates by algorithm. All edges updated by the algorithm lie in a sub-graph (forest) with l leaves and hopping diameter at most k. In that sub-graph, by Lemma 9.9, there are at most $\left\lfloor \frac{k}{2} \right\rfloor \cdot l + (k \mod 2)$ edges.

Comparison. So compared to OPT the algorithm made at most $\left\lfloor \frac{k}{2} \right\rfloor \cdot l + (k \mod 2) = \left\lfloor \frac{k}{2} \right\rfloor \cdot (l - 2) + k + (k \mod 2) \leq \left\lfloor \frac{k}{2} \right\rfloor \cdot OPT + k + (k \mod 2)$ updates.

□

Performance of algorithm in other scenarios. The algorithm relies on the property we have established with Lemma 9.10 to achieve the upper bound we have shown in Lemma 9.15. The proof of Lemma 9.10 is based on the fact that between any two vertices there is a unique path, and further makes use of the restriction of each uncertainty area being either trivial or an open interval. One would assume that Lemma 9.10 does not hold without those settings, nor for arbitrary graphs. Therefore it is expected that our algorithm does not perform the same under other settings. We give the following two examples for justification.

Example 1. Lets consider the edge uncertainty graph $\mathcal{U} = (V, E, A)$ in Figure 9.5, which is a star with $V = \{g, h, p, q\}$. $\mathcal{U}$ consists of one edge with non-trivial uncertainty area value $[1, 3]$ and two more edges with trivial uncertainty areas values $\{4\}$ and $\{3\}$. Here, for every weighted graph $G = (V, E, w)$ that is consistent with $\mathcal{U}$, we can see that $d_w(g, p) \geq d_w(g, q)$ and $d_w(g, p) > d_w(p, q)$ and therefore the problem is solved without any updates with (g, p) being a farthest pair of vertices in G. However as the update of the edge with uncertainty area value $[1, 3]$ allows both actual weights $\{3\}$ and $\{2\}$, there exist two weighted graphs $G' = (V, E, w')$ and $G'' = (V, E, w'')$ consistent with $\mathcal{U}$ such that $d_{w'}(g, p) = d_{w'}(g, q) = 7$ and $d_{w''}(g, p) = 7$, $d_{w''}(g, q) = 6$ and so $d_{w''}(g, p) > d_{w''}(g, q)$. Therefore (g, p) and (g, q) are both farthest pairs of vertices in G' but not in G''. Hence Lemma 9.10 does not hold if closed intervals are allowed as uncertainty areas.

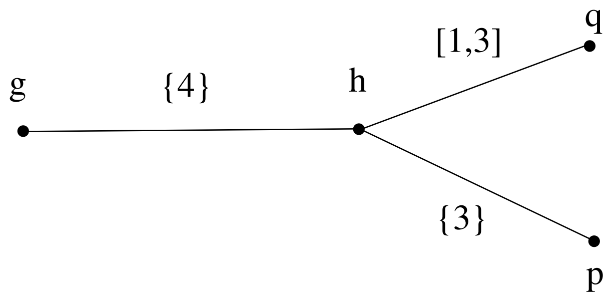


Figure 9.5: Tree with a closed interval

Example 2. Lets consider the edge uncertainty graph $\mathcal{U} = (V, E, A)$ in Figure 9.6, which contains a cycle. $\mathcal{U}$ contains in its cycle an edge with the non-trivial uncertainty area value $(2, 10)$. Let e be this edge and so $A_e = (2, 10)$. We note that in case the actual weight of e is more than 8 then the alternative path between h and r would be cheaper. Therefore the distance between g and q in every weighted graph $G = (V, E, w)$ that is consistent with $\mathcal{U}$ cannot be more than $20 + 8 + 1 + 1 = 30$. With similar arguments as in Example 1, since $d_w(g, p) = 30$, we can see that (g, p) is a farthest pair of vertices in G and the problem is solved without any updates. Furthermore, as the update of the edge e allows both actual weights $\{8\}$ and $\{7\}$, there exist two weighted graphs $G' = (V, E, w')$ and $G'' = (V, E, w'')$ consistent with $\mathcal{U}$ such that $d_{w'}(g, p) = d_{w'}(g, q) = 30$ and $d_{w''}(g, p) = 30$, $d_{w''}(g, q) = 29$ and so $d_{w''}(g, p) > d_{w''}(g, q)$. Therefore (g, p) and (g, q) are both farthest pairs of vertices in G' but not in G''. Hence Lemma 9.10 does not hold in graphs where between two vertices there can exist multiple paths, even if all uncertainty areas are either trivial or open intervals.

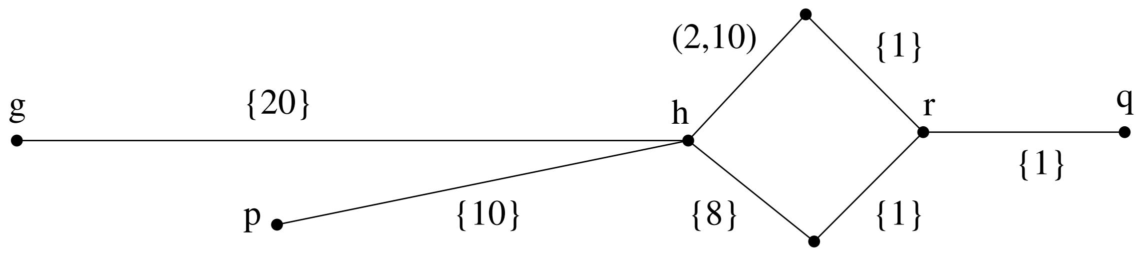


Figure 9.6: Graph with a cycle

9.3.1.2 Lower Bound

For the lower bound of this subsection we construct a non-solved edge uncertainty graph $\mathcal{U} = (V, E, A)$ of the FPVT problem and compare the number of updates any deterministic online algorithm is forced to perform against OPT. Recall that in this subsection we assume that every uncertainty area is either trivial or an open interval. We will only use the two different open intervals $(1, 2)$ and $(2 - \frac{1}{2n}, 4)$ for all uncertainty areas of E and three different actual weights which we give later on. For convenience we say an edge $e \in E$ is either of *type L* if $A_e = (1, 2)$ or of *type H* if $A_e = (2 - \frac{1}{2n}, 4)$.

For the construction we take $n \geq 2$ as a parameter for the number of consecutive edges in every distinct branch of the graph, and $r \geq 2$ for the number of branches that consists of type L edges.

We start by placing $2n$ type H edges consecutively, and let m be the vertex in the middle. Then we place r number of distinct branches of edges, where each branch consists of n type L edges and let all these branches share vertex m as one of their endpoints. This construction makes the hopping diameter to be $k = 2n$. We note that this construction forms a generalised version of a star, where each branch contains n edges instead of just 1. See Figure 9.7 for the example of $n = 3$ and $r = 4$.

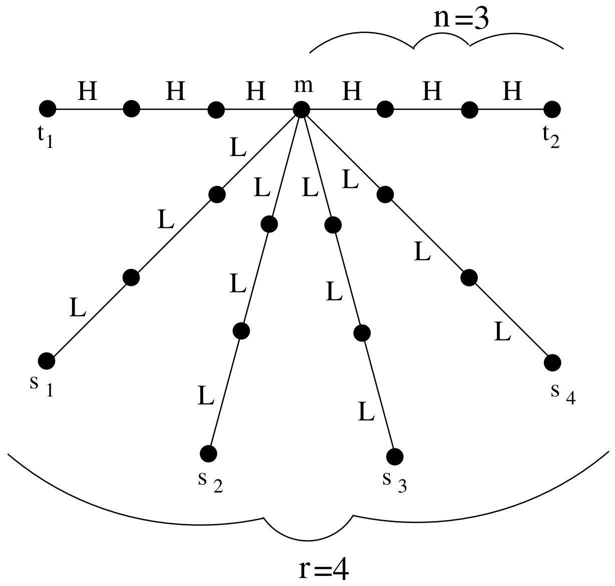


Figure 9.7: Construction with type L and type H edges

Next we give the details about the three actual weights used. Let $a = 1 + \frac{1}{4n}$, $b = 2 - \frac{1}{4n}$ and $c = 3$. We note that for each edge $e \in E$ if it is of type L then only $a, b \in A_e$, or if it is of type H then only $b, c \in A_e$. Furthermore $a < b < c$.

So the information of an uncertainty area is the same for all type L and the same for all type H edges, however the precise information obtained with an update operation varies depending in the order an online strategy chooses the updates. An uncertainty area is reduced to contain only an actual weight given by the adversary according to Algorithm 7:

Algorithm 7 Adversary for the lower bound of FPVT

```
if Type $L$ edge then
    if Last available update in the construction then
        Receive $a$;
    else
        if Last update on current branch then
            if Last update of all type $L$ edges then
                Receive $b$;
            else
                Receive $a$;
            end if
        else
            Receive $b$;
        end if
    end if
else if Type $H$ edge then
    if Last available update in the construction then
        Receive $c$;
    else
        if Last update on current branch then
            if Last update of all type $H$ edges then
                Receive $b$;
            else
                Receive $c$;
            end if
        else
            Receive $b$;
        end if
    end if
end if
```

Following the reasoning of how the result of an update is given we have the following property:

Lemma 9.16. *Let $\mathcal{U}'$ be the edge uncertainty graph after updating every edge in $\mathcal{U}$ in any order, with actual weights obtained as determined by Algorithm 7. Then exactly one of the following holds for $\mathcal{U}''$:*

a)In every branch i of type L edges and leaf s_i, there exists exactly one edge $e \in P(s_i, m)$ with trivial uncertainty area $A_e = \{a\}$.

b)In every branch j of type H edges and leaf t_j, there exists exactly one edge $e \in P(t_j, m)$ with trivial uncertainty area $A_e = \{c\}$.

Proof. We argue that at least a) or b) condition holds. Either the last update in the construction was made for a type L edge or for a type H. First we assume it was for type L. Then line 3 of the adversary must have occurred once at the last branch considered and line 9 must have occurred once for every branch before that (following the conditions in the previous lines). Therefore each branch of type L edges contains an edge with uncertainty area $\{a\}$. Similarly, if the last update in the construction was made for a type H edge, then line 17 must have occurred once at the last branch considered and line 23 must have occurred once for every branch before that. Therefore each branch of type H edges contains an edge with uncertainty area $\{c\}$. So either a) or b) must be true.

To show that not both conditions a) and b) can be true we argue that there exist a branch that consists of only edges with uncertainty area $\{b\}$. In case the last update made in the construction was for a type L edge, then that means line 17 cannot have occurred before and the only line other than 17 which yields a $\{c\}$ update is line 23. As line 23 cannot occur for the last update of all type H edges, and as it only occurs for the last in a particular branch update of type H edges, then there must be a branch of type H edges that consists of only edges with uncertainty area $\{b\}$. Similarly if the last update made in the construction was for a type H edge instead, line 3 cannot have occurred before and the only line other than 3 which yields an $\{a\}$ update is line 9. As line 9 cannot occur for the last update of all type L edges, and as it only occurs for the last in a particular branch update of type L edges, then there must be a branch of type L edges that consists of only edges with uncertainty area $\{b\}$. So not both a) and b) can be true.

□

The construction has the following property:

Lemma 9.17. *After updating every edge in $\mathcal{U}$ in any order, with actual weights obtained as determined by Algorithm 7, the pair (t_1, t_2) is the only farthest pair of vertices in that weighted graph.*

Proof. Let $\mathcal{U}' = (V, E, A')$ be the edge uncertainty graph after every edge in $\mathcal{U}$ has been updated. We note that since for every edge in $\mathcal{U}'$ the uncertainty area is trivial, there exists only one weighted graph (V, E, w') that is consistent with $\mathcal{U}'$.

By Lemma 9.16, in $\mathcal{U}'$ either (Case 1:) $P(t_1, t_2)$ contains $2n - 1$ edges with uncertainty area value $\{b\}$ and one more with $\{c\}$, or (Case 2:) $P(t_1, t_2)$ contains $2n-2$ edges with uncertainty area value $\{b\}$ and two more with $\{c\}$. So the distance between t_1 and t_2 is:

Case 1: $d_{w'}(t_1, t_2) = (2n-1)(2-\frac{1}{4n})+3 = 4n-\frac{2}{4}-2+\frac{1}{4n}+3 = 4n+\frac{1}{4n}+\frac{2}{4}$

Case 2: $d_{w'}(t_1, t_2) = (2n-2)(2-\frac{1}{4n})+2(3) = 4n-\frac{2}{4}-4+\frac{2}{4n}+6 = 4n+\frac{2}{4n}+\frac{6}{4}$

See also Figure 9.8 and Figure 9.9:

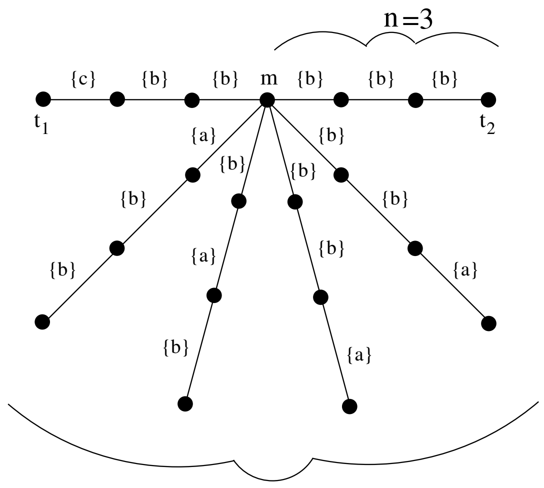


Figure 9.8: Example of Case 1

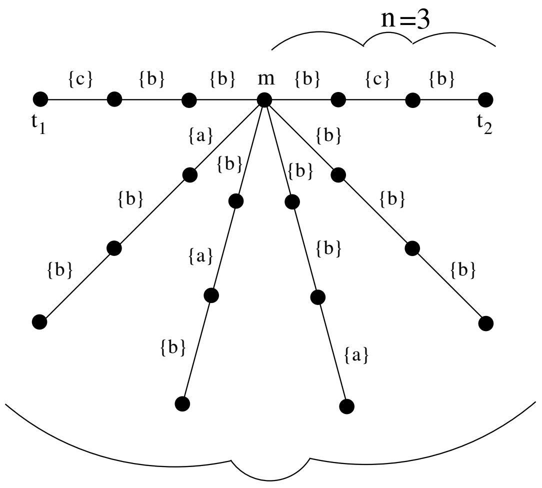


Figure 9.9: Example of Case 2

Next let $x, y \in V$ be two leaves in the construction such that either $x \notin \{t_1, t_2\}$ or $y \notin \{t_1, t_2\}$. For $\mathcal{U}'$, as by the construction, $P(x, y)$ contains at least $2n - 2$ edges with uncertainty area value $\{b\}$. Further $P(x, y)$ contains two more which we note are dependent to what $P(t_1, t_2)$ contains. We separate the 6 cases according to what the uncertainty areas of the remaining two edges are, when combined with our previous cases. Let the remaining two edges have the uncertainty area values:

Case 1.1: $\{a\}$ and $\{a\}$. So

$d_{w'}(x, y) = (2n-2)(2-\frac{1}{4n})+2(1+\frac{1}{4n}) = 4n-\frac{2}{4}-4+\frac{2}{4n}+2+\frac{2}{4n} = 4n+\frac{1}{n}-\frac{10}{4}$

And as $4n + \frac{1}{n} - \frac{10}{4} < 4n + \frac{1}{4n} + \frac{2}{4}$ then, by Definition 9.2, (x, y) cannot be a farthest pair of vertices.

Case 1.2: $\{a\}$ and $\{b\}$. So

$d_{w'}(x, y) = (2n-1)(2-\frac{1}{4n})+1+\frac{1}{4n} = 4n-\frac{2}{4}-2+\frac{1}{4n}+1+\frac{1}{4n} = 4n+\frac{2}{4n}-\frac{6}{4}$

And as $4n + \frac{2}{4n} - \frac{6}{4} < 4n + \frac{1}{4n} + \frac{2}{4}$ then, by Definition 9.2, (x, y) cannot be a farthest pair of vertices.

Case 1.3: $\{a\}$ and $\{c\}$. So

$d_{w'}(x, y) = (2n-2)(2-\frac{1}{4n})+1+\frac{1}{4n}+3 = 4n-\frac{2}{4}-4+\frac{2}{4n}+1+\frac{1}{4n}+3 = 4n+\frac{3}{4n}-\frac{2}{4}$

And as $4n + \frac{3}{4n} - \frac{2}{4} < 4n + \frac{1}{4n} + \frac{2}{4}$ then, by Definition 9.2, (x, y) cannot be a farthest pair of vertices.

Case 2.1: $\{a\}$ and $\{b\}$. So

$d_{w'}(x, y) = (2n-1)(2-\frac{1}{4n})+1+\frac{1}{4n} = 4n-\frac{2}{4}-2+\frac{1}{4n}+1+\frac{1}{4n} = 4n+\frac{2}{4n}-\frac{6}{4}$

And as $4n + \frac{2}{4n} - \frac{6}{4} < 4n + \frac{2}{4n} + \frac{6}{4}$ then, by Definition 9.2, (x, y) cannot be a farthest pair of vertices.

Case 2.2: $\{a\}$ and $\{c\}$. So

$d_{w'}(x, y) = (2n-2)(2-\frac{1}{4n})+1+\frac{1}{4n}+3 = 4n-\frac{2}{4}-4+\frac{2}{4n}+1+\frac{1}{4n}+3 = 4n+\frac{3}{4n}-\frac{2}{4}$

And as $4n + \frac{3}{4n} - \frac{2}{4} < 4n + \frac{2}{4n} + \frac{6}{4}$ then, by Definition 9.2, (x, y) cannot be a farthest pair of vertices.

Case 2.3: $\{b\}$ and $\{c\}$. So
$d_{w'}(x, y) \ = \ (2n - 1)(2 - \frac{1}{4n}) + 3 \ = \ 4n - \frac{2}{4} - 2 + \frac{1}{4n} + 3 \ = \ 4n + \frac{1}{4n} + \frac{2}{4}$
And as $4n + \frac{1}{4n} + \frac{2}{4} < 4n + \frac{2}{4n} + \frac{6}{4}$ then, by Definition 9.2, (x, y) cannot be a farthest pair of vertices.

Following our case analysis, (t_1, t_2) is the only farthest pair of vertices for $\mathcal{U}'$.

□

Let s_i be the leaf for the ith branch of type L edges for every $0 < i \leq r$, and let t_1, t_2 be the leaves for the two branches of type H edges. We note the following uncertainty distances of the construction:
$d_A(s_i, m) \ = \ (n, 2n)$
$d_A(t_j, m) \ = \ (n(2 - \frac{1}{2n}), 4n) \ = \ (2n - \frac{1}{2}, 4n)$
This brings to the following:

Lemma 9.18. *Let $\mathcal{U}' = (V, E, A')$ be an edge uncertainty graph after updating any number of edges in $\mathcal{U}$ in any order, with actual weights obtained as determined by Algorithm 7. Then $\mathcal{U}'$ is solved if and only if $d_{A'}(t_j, m) \geq d_{A'}(s_i, m)$ for every i and j.*

Proof. We first note $d_{A'}(t_j, m) \cap d_{A'}(s_i, m) = \emptyset$.

Following our case analysis in Lemma 9.17, the pair (t_1, t_2) is the only farthest pair of vertices after updating every edge in $\mathcal{U}$. This means that if $\mathcal{U}'$ is solved then in every weighted graph (V, E, w') that is consistent with $\mathcal{U}'$, the pair (t_1, t_2) is a farthest pair of vertices and therefore $d_{w'}(t_1, t_2) \geq d_{w'}(x, y)$ for every $x, y \in V$. If this is the case then, as $d_{w'}(t_1, t_2) = d_{w'}(t_1, m) + d_{w'}(t_2, m)$, also $d_{w'}(t_j, m) \geq d_{w'}(s_i, m)$. Hence if $\mathcal{U}'$ is solved then $d_{A'}(t_j, m) \geq d_{A'}(s_i, m)$ holds.

If $d_{A'}(t_j, m) \geq d_{A'}(s_i, m)$ then in every weighted graph (V, E, w'') that is consistent with $\mathcal{U}'$ we have $d_{w''}(t_j, m) \geq d_{w''}(s_i, m)$ and as $d_{w''}(t_1, m) + d_{w''}(t_2, m) = d_{w''}(t_1, t_2)$ then also $d_{w''}(t_1, t_2) \geq d_{w''}(s_i, t_j) \geq d_{w''}(s_i, s_q)$ for every $0 < q \leq r$. If this is the case then the pair (t_1, t_2) is a farthest pair of vertices for (V, E, w''), and hence if $d_{A'}(t_j, m) \geq d_{A'}(s_i, m)$ then $\mathcal{U}'$ is solved.

□

We now show:

Lemma 9.19. *After $(r+2)n-1$ updates on $\mathcal{U}$ by any deterministic online algorithm with actual weights obtained as determined by Algorithm 7, the edge uncertainty graph is not solved.*

Proof. Following the proof of Lemma 9.16, every branch of type L edges except one contains one edge with uncertainty area $\{a\}$, and every branch of type H edges except one contains one edge with uncertainty area $\{c\}$.

Let $\mathcal{U}' = (V, E, A')$ be the edge uncertainty graph after these $(r+2)n-1$ updates have been made on $\mathcal{U}$. As E consists of exactly $(r+2)n$ edges, if $(r+2)n-1$ updates have been made then there is exactly one edge that has not been updated. Let the non-updated edge be e' and we note that either e' is of type L (Case 1) or e' is of type H (Case 2).

Case 1: As e' is of type L and not been updated then $A_{e'} = (1, 2)$. Let s_1 be the leaf of this particular branch that contains e'. This branch further contains $n-1$ edges with the trivial uncertainty area $\{b\}$ and we note its uncertainty distance: $d_{A'}(s_1, m) = (n-1)(2-\frac{1}{4n})+(1,2) = 2n+\frac{1}{4n}-\frac{9}{4}+(1,2) = (2n+\frac{1}{4n}-\frac{5}{4}\ ,\ 2n+\frac{1}{4n}-\frac{1}{4})$.

Further let t_1 be the leaf of the branch of type H edges that does not contain an edge with uncertainty area $\{c\}$ and we note its uncertainty distance:

$d_{A'}(t_1, m) = n(2 - \frac{1}{4n}) = [2n - \frac{1}{4}\ ,\ 2n - \frac{1}{4}]$.

Since $[2n - \frac{1}{4}\ ,\ 2n - \frac{1}{4}] \not\geq (2n + \frac{1}{4n} - \frac{5}{4}\ ,\ 2n + \frac{1}{4n} - \frac{1}{4})$ then, by Lemma 9.18, $\mathcal{U}'$ is not solved.

Case 2: As e' is of type H and not been updated then $A_{e'} = (2 - \frac{1}{2n}, 4)$. Let t_1 be the leaf of this particular branch that contains e'. This branch further contains $n-1$ edges with the trivial uncertainty area $\{b\}$ and we note its uncertainty distance: $d_{A'}(t_1, m) = (n-1)(2 - \frac{1}{4n}) + (2 - \frac{1}{2n}, 4) = 2n + \frac{1}{4n} - \frac{9}{4} + (2 - \frac{1}{2n}, 4) = (2n - \frac{1}{4n} - \frac{1}{4}\ ,\ 2n + \frac{1}{4n} + \frac{7}{4})$.

Further let s_1 be the leaf of the branch of type L edges that does not contain an edge with uncertainty area $\{a\}$ and we note its uncertainty distance:

$d_{A'}(s_1, m) = n(2 - \frac{1}{4n}) = [2n - \frac{1}{4}\ ,\ 2n - \frac{1}{4}]$.

Since $(2n - \frac{1}{4n} - \frac{1}{4}\ ,\ 2n + \frac{1}{4n} + \frac{7}{4}) \not\geq [2n - \frac{1}{4}\ ,\ 2n - \frac{1}{4}]$ then, by Lemma 9.18, $\mathcal{U}'$ is not solved.

□

Lemma 9.20. *OPT is at most r, for the edge uncertainty graph $\mathcal{U}$ with actual weights obtained as determined by Algorithm 7.*

Proof. By Lemma 9.19 every online strategy is forced to update $(r + 2)n$ to reach a solved instance, and by Lemma 9.16 after $(r + 2)n$ updates either every branch of type L edges contains exactly one edge with uncertainty area $\{a\}$, or every branch of type H edges contains exactly one edge with uncertainty area $\{c\}$.

We argue that OPT is either the number of all edges that have the uncertainty area $\{a\}$, if there exists one in every branch of type L edges, or the number of all edges that have the uncertainty area $\{c\}$, if there exists one in every branch of type

H edges. As there are exactly 2 branches for type H edges and $r \geq 2$ branches of type L edges then it follows $OPT \leq r$.

It remains to show that if $\mathcal{U}' = (V, E, A')$ is the edge uncertainty graph after updating these edges in $\mathcal{U}$, even if all other edges are not updated, $\mathcal{U}'$ is solved. By Lemma 9.18, it suffices to show that the condition $d_{A'}(t_j, m) \geq d_{A'}(s_i, m)$ holds for every i and j.

We split the two cases to compare uncertainty distances for where there exists one uncertainty area with value $\{a\}$ in every branch of type L edges (Case 1, see also Figure 9.8), and where there exists one uncertainty area with value $\{c\}$ in every branch of type H edges (Case 2, see also Figure 9.9).

Case 1: Each branch j of type H edges has the same uncertainty distance from leaf to vertex m as it is composed of n non-updated edges with uncertainty area $(2 - \frac{1}{2n}\ ,\ 4)$. So $d_{A'}(t_j, m) \ = \ n(2 - \frac{1}{2n}\ ,\ 4) \ = \ (2n - \frac{1}{2}\ ,\ 4n)$.
Further each branch i of type L edges has the same uncertainty distance from leaf to vertex m as it is composed of $n-1$ non-updated edges with uncertainty area $(1, 2)$ and one more edge that has been updated and now has the trivial uncertainty area $\{1 + \frac{1}{4n}\}$. So $d_{A'}(s_i, m) \ = \ (n-1)(1\ ,\ 2) + (1 + \frac{1}{4n}) \ = \ (n-1\ ,\ 2n-2) + (1 + \frac{1}{4n}) \ = \ (n + \frac{1}{4n}\ ,\ 2n + \frac{1}{4n} - 1)$.
As $(2n - \frac{1}{2}\ ,\ 4n) \ > \ (n + \frac{1}{4n}\ ,\ 2n + \frac{1}{4n} - 1)$ for any n then condition $d_{A'}(t_j, m) \geq d_{A'}(s_i, m)$ holds.

Case 2: Each branch j of type H edges has the same uncertainty distance from leaf to vertex m as it is composed of $n-1$ non-updated edges with uncertainty area $(2 - \frac{1}{2n}\ ,\ 4)$ and one more edge that has been updated and now has the trivial uncertainty area $\{3\}$. So $d_{A'}(t_j, m) \ = \ (n-1)(2 - \frac{1}{2n}\ ,\ 4) + (3) \ = \ (2n - \frac{1}{2} - 2 + \frac{1}{2n}\ ,\ 4n - 4) + 3 \ = \ (2n + \frac{1}{2n} + \frac{1}{2}\ ,\ 4n - 1)$.
Further each branch i of type L edges has the same uncertainty distance from leaf to vertex m as it is composed of n non-updated edges with uncertainty area $(1, 2)$. So $d_{A'}(s_i, m) \ = \ n(1\ ,\ 2) \ = \ (n\ ,\ 2n)$.
As $(2n + \frac{1}{2n} + \frac{1}{2}\ ,\ 4n-1) \ > \ (n\ ,\ 2n)$ for any n then condition $d_{A'}(t_j, m) \geq d_{A'}(s_i, m)$ holds.

□

We now use the established Lemmas to show the desired lower bound for FPVT:

Lemma 9.21. *Under the restriction of open or trivial uncertainty areas, there exist configurations of the FPVT problem such that every deterministic online algorithm performs* $\lfloor \frac{k}{2} \rfloor \cdot OPT + (k \mod 2) + k$ *updates to solve the problem, where* k *is the hopping diameter.*

Proof. By Lemma 9.19 every online strategy requires $nr + 2n$ updates to solve an FPVT problem as by our construction, and by Lemma 9.20 the optimal update solution of the same instance contains at most r updates. Therefore we get a lower bound of $n \cdot OPT + 2n$. Furthermore, as the hopping diameter is $k = 2n$ for that construction, the lower bound is also bounded by k so $\frac{k}{2} \cdot OPT + k$. Moreover as k is even for any n then $\frac{k}{2} = \lfloor \frac{k}{2} \rfloor$ and $(k \mod 2) = 0$, and therefore $\frac{k}{2} \cdot OPT + k = \lfloor \frac{k}{2} \rfloor \cdot OPT + (k \mod 2) + k$.

□

9.3.1.3 Conclusion

Following Lemma 9.15 and Lemma 9.21, we summarise our results for this subsection with Theorem 9.22:

Theorem 9.22. *Under the restriction of open or trivial uncertainty areas, the witness algorithm for the FPVT problem performs at most $\lfloor \frac{k}{2} \rfloor \cdot OPT + (k \mod 2) + k$ updates, where k is the hopping diameter. Furthermore, this is the best possible.*

Note. Following Lemma 9.10, for a solved edge uncertainty graph $\mathcal{U}$, all farthest pairs of vertices in weighted graphs that are consistent with $\mathcal{U}$ are computable without further updates. Therefore we note Theorem 9.22 must carry over to the variation of the FPVT problem where not just one but all farthest pairs of vertices need to be computable.

9.3.2 Closed Intervals

Introduction. We now look at the FPVT problem without restricting the uncertainty areas to be either trivial or open intervals. Specifically we show with a construction of an edge uncertainty graph $\mathcal{U} = (V, E, A)$ that there does not exist a deterministic online algorithm with bounding function over OPT, using uncertainty areas which are either trivial or closed intervals.

For this construction we first place an edge with trivial uncertainty area containing a large weight such that one vertex is the leaf x and the other vertex m is connected to n branches where each branch contains n consecutive edges. Let all these $n \cdot n$ edges have the uncertainty area $[1, 2]$ and we can fix the large weight to be $\{2n+1\}$. Further let $Y = \{y_1, y_2, \ldots, y_n\}$ denote the set of all leaves of these $n \cdot n$ edges. So the constructed tree contains $1 + n$ leaves and $n \cdot n$ edges with non-trivial uncertainty areas. We denote the constructed uncertainty graph throughout this subsection by $\mathcal{U} = (V, E, A)$. See Figure 9.10.

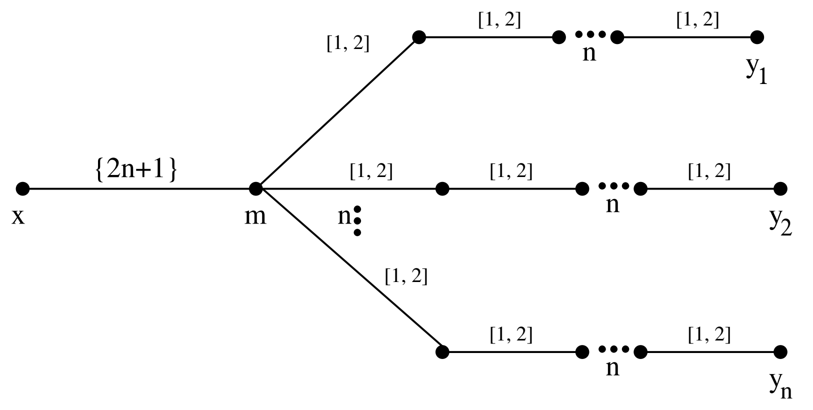


Figure 9.10: Construction of lower bound for FPVT

We highlight the properties of the construction before any updates are considered. For every weighted graph $G = (V, E, w)$ that is consistent with $\mathcal{U}$, we have $d_w(x, m) > d_w(y_i, m)$ for every $0 < i \leq n$. So we note the following:

$$\begin{aligned} d_A(x, m) &= \{2n + 1\} \\ d_A(y_i, m) &= (n, 2n) \\ d_A(x, m) &> d_A(y_i, m) \end{aligned}$$

As such x must be a vertex in every farthest pair of vertices in every weighted graph that is consistent with $\mathcal{U}$. Thus, in order to determine a farthest pair of vertices for every weighted graph that is consistent with $\mathcal{U}$, it remains to find a vertex y_j for some $0 < j \leq n$. However, with the uncertainty distance $d_A(y_i, m)$ being the same for every i, there exist weighted graphs that are consistent with $\mathcal{U}$ such that each vertex in Y is inside and not inside a farthest pair of vertices. Therefore $\mathcal{U}$ is not solved.

These properties bring us to the following Lemma:

Lemma 9.23. *Let $\mathcal{U}' = (V, E, A')$ be an edge uncertainty graph after updating any number of edges in $\mathcal{U}$ in any order. Then $\mathcal{U}'$ is solved if and only if there exists $y_j \in Y$ such that $d_{A'}(m, y_j) \geq d_{A'}(m, y_k)$ for every $y_k \in Y \setminus \{y_j\}$.*

Proof. As in every weighted graph that is consistent with $\mathcal{U}$ the vertex x is inside every farthest pair of vertices, then this must also be the case for $\mathcal{U}'$. Further we have the set Y contains all leaves that are not x.

So in order for $\mathcal{U}'$ to be solved, there must exist a pair (x, y_z) for some $y_z \in Y$ such that for every weighted graph (V, E, w') that is consistent with $\mathcal{U}'$ then $d_{w'}(x, y_z) \geq d_{w'}(a, b)$ for every $a, b \in V$. Since the sub-path between x and m is shared among all possible farthest pairs and furthermore $P(m, y_j) \cap P(m, y_k) = \emptyset$ then it suffices to show $d_{A'}(m, y_j) \geq d_{A'}(m, y_k)$.

With similar arguments if $\mathcal{U}'$ is solved then $d_{A'}(m, y_j) \geq d_{A'}(m, y_k)$ holds.

□

The idea behind the construction is to force any deterministic online algorithm to perform $n \cdot n$ updates (i.e. update every non-trivial uncertainty area in the construction) where OPT is only n. To achieve this, when an algorithm requests to update an edge, an uncertainty area is reduced to contain only an actual weight given by the adversary according to Algorithm 8:

Algorithm 8 Adversary for the lower bound of FPVT with closed intervals allowed

```
1: if Last available update in the construction then
2:     Receive 2;
3: else
4:     if Last update on current branch then
5:         Receive 1.1;
6:     else
7:         Receive 2;
8:     end if
9: end if
```

We look at further properties of the construction in the following two lemmas.

Lemma 9.24. *After $n \cdot n - 1$ updates on $\mathcal{U}$ by any deterministic online algorithm with actual weights obtained as determined by Algorithm 8, the edge uncertainty graph is not solved.*

Proof. Let $\mathcal{U}' = (V, E, A')$ be the edge uncertainty graph after $n \cdot n - 1$ updates have been made on $\mathcal{U}$. Graph $\mathcal{U}$ contains exactly $n \cdot n$ edges with non-trivial uncertainty areas, so if $n \cdot n - 1$ updates have been made there exists exactly one edge e with non-trivial uncertainty area in $\mathcal{U}'$. So, as by Algorithm 8, every branch (other than $P(x, m)$) contains $n - 1$ edges with uncertainty area value $\{2\}$ and further either contains the trivial uncertainty area with value $\{1.1\}$ or the non-updated edge e with uncertainty area $A'_e = [1, 2]$. Let y_1 be the leaf of a branch for the former, and let y_2 be the leaf of the branch for the latter. The uncertainty distances are as follows:

$$d_{A'}(m, y_1) = (n-1)2 + 1.1 = 2n - 2 + 1.1 = [2n - 0.9\ ,\ 2n - 0.9]$$
$$d_{A'}(m, y_2) = (n-1)2 + [1, 2] = 2n - 2 + [1, 2] = [2n - 1\ ,\ 2n]$$

So, as $[2n-1\ ,\ 2n] \not\gtrsim [2n-0.9\ ,\ 2n-0.9]$ and $[2n-0.9\ ,\ 2n-0.9] \not\gtrsim [2n-1\ ,\ 2n]$, the condition of Lemma 9.23 does not hold and therefore $\mathcal{U}'$ is not solved. □

Lemma 9.25. *OPT is at most n, for the edge uncertainty graph $\mathcal{U}$ with actual weights obtained as determined by Algorithm 8.*

Proof. As by the construction there are n branches with leaves in Y. Furthermore exactly one of those branches consists only of edges which when updated their area of uncertainty will be reduced to $\{2\}$, and let the leaf for this branch be $y_1 \in Y$.

Moreover the rest $n-1$ branches consist of $n-1$ edges which when updated their area of uncertainty will be reduced to $\{2\}$ and for the remaining edge to $\{1.1\}$.

We argue that $P(m, y_1)$ is an optimal update solution by showing that after updating all edges in $P(m, y_1)$ the instance is solved with $|P(m, y_1)| = n$ updates.

Let $\mathcal{U}' = (V, E, A')$ be the edge uncertainty graph after updating in $\mathcal{U}$ every edge in $P(m, y_1)$. The uncertainty distance between m and y_1 becomes:

$d_{A'}(m, y_1) \ = \ n(2) = [2n \ , \ 2n]$

And the uncertainty distance between m and y_i for every $y_i \in Y \setminus \{y_1\}$ remains:

$d_{A'}(m, y_i) \ = \ n[1, 2] \ = \ [n \ , \ 2n]$

As $[2n \ , \ 2n] \ \geq \ [n \ , \ 2n]$, by Lemma 9.23, $\mathcal{U}'$ is solved with n updates.

□

Following Lemma 9.24 and Lemma 9.25, any online strategy can be forced to make $n \cdot n$ updates to reach a solved instance where, if the results of the updates were known beforehand, the problem could have been solved with just n. This brings us to the following conclusion for this subsection:

Theorem 9.26. *If closed intervals are allowed as uncertainty areas, there does not exist a deterministic online algorithm for the FPVT problem with constant update competitive ratio.*

It remains however open if the problem was to determine all farthest pairs of vertices instead of just one.

9.4 FPV for Cycle Graphs

Introduction. In this section we give a lower bound for finding a farthest pair of vertices on a different kind of graphs. The setting is very similar to Section 9.3 with the difference being that now there is a single cycle through all vertices, and so there are exactly two paths between any two vertices instead of just one. We use the abbreviation FPVC for the Farthest Pair of Vertices in a Cycle graph under uncertainty, which is in line with Definition 9.3. We will make a construction of an edge uncertainty graph and show that any deterministic online algorithm has to update all n edges with non-trivial uncertainty area, whereas OPT is just k where $n \geq k \geq 1$.

We modify Definition 9.7 to be used for cycle graphs:

Definition 9.27. *For a set S of consecutive edges in an edge uncertainty graph $\mathcal{U}' = (V, E, A')$, we say the sum of all uncertainty areas of the edges in S is the uncertainty distance of S. We write $d_{A'}(S)$.*

And we note the uncertainty distance of a set S is either in the form of a closed interval, if all edges in S have uncertainty areas that are trivial or closed intervals, or in the form of an open interval, if at least one of the edges in S has its uncertainty area to be an open interval. We also note that we use this notation for comparison between two edge sets only when they do not share an element.

Placement of edges and vertices. For this construction we first take $n \geq 1$ as a parameter and construct the set of edges $H = \{h^1, h^2, h^3, \ldots, h^n\}$ such that each edge $h^i \in H$ has the non-trivial uncertainty area $A_{h^i} = (1, 3)$ and placed consecutively forming a horizontal line. Let s_1 and t_1 be the leaves of this line. Now we place the edge l with trivial uncertainty area $A_l = \{n + 1\}$ below and parallel to the edges in H, without sharing any edges or vertices with them. Let s_2 and t_2 be the leaves of this edge (without loss of generality t_2 is closer to t_1 than s_2 to t_1). Next step is to connect t_1 with t_2 forming the edge q which has the trivial uncertainty area $A_q = \{4n\}$. Finally we connect vertices s_1 and s_2 with two more edges g^1 and g^2 respectively meeting on vertex m. The uncertainty area for each of these two edges is trivial and set to be $A_{g^1} = A_{g^2} = \{10n\}$.

The construction of the edge uncertainty graph in now complete and we denote it throughout this section with $\mathcal{U} = (V, E, A)$. The construction can be seen in Figure 9.11. The set E contains exactly 4 edges with trivial uncertainty area and further n edges with non-trivial uncertainty area.

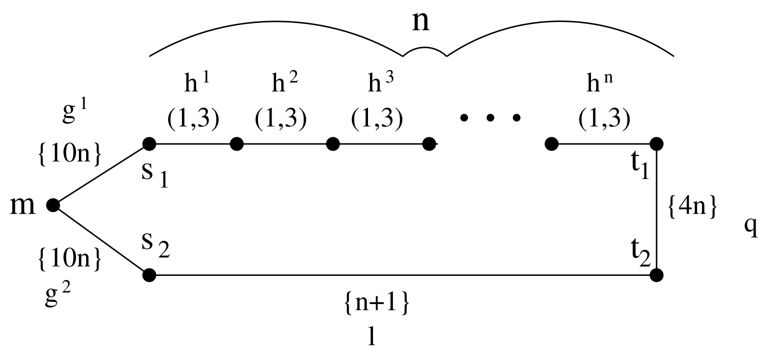


Figure 9.11: Construction of lower bound for FPVC

We highlight some properties before any updates are considered. For every weighted graph $G = (V, E, w)$ that is consistent with $\mathcal{U}$:

- Vertex m is inside every farthest pair of vertices in G, as the distance to m from any other vertex is at least $10n$, whereas the distance between any other two vertices must be less. For example from s_1 to s_2 there are two paths: one with visiting m and weight $20n$, and the alternative path cannot be more than $n(1,3) + (4n) + (n+1)$. So as $10n > n(1,3) + (4n) + (n+1)$ then $10n > d_w(s_1, s_2)$.
- Since $4n > n(1,3)$ and $4n > n+1$, it follows that $d_w(t_1, t_2) > d_w(s_1, t_1)$ and $d_w(t_1, t_2) > d_w(s_2, t_2)$. So there exists a path from vertex m to every other vertex such that visiting edge q is more expensive than the alternative path.

Combining the above properties, we have that (m, t_1) and (m, t_2) are the only potential farthest pairs of vertices for weighted graphs consistent with $\mathcal{U}$. This brings us to the following Lemma:

Lemma 9.28. *Let $\mathcal{U}' = (V, E, A')$ be an edge uncertainty graph after updating any number of edges in $\mathcal{U}$ in any order. Then $\mathcal{U}'$ is solved if and only if either $d_{A'}(H) \geq d_{A'}(\{l\})$ or $d_{A'}(\{l\}) \geq d_{A'}(H)$.*

Proof. The last property we have highlighted implies that $\mathcal{U}'$ is solved if and only if for every weighted graph (V, E, w') that is consistent with $\mathcal{U}'$, either $d_{w'}(m, t_1) \geq d_{w'}(m, t_2)$ or $d_{w'}(m, t_2) \geq d_{w'}(m, t_1)$.

Furthermore in the properties it is shown that $d_{w'}(m, t_1)$ must contain the weight of edge g^1 and $d_{w'}(m, t_2)$ must contain the weight of edge g^2. Therefore, as $A_{g^1} = A_{g^2} = \{10n\}$ for $\mathcal{U}$ and also for $\mathcal{U}'$, it follows that $\mathcal{U}'$ is solved if and only if either $d_{w'}(s_1, t_1) \geq d_{w'}(s_2, t_2)$ or $d_{w'}(s_2, t_2) \geq d_{w'}(s_1, t_1)$.

This further implies that $\mathcal{U}'$ is solved if and only if either $d_{A'}(H) \geq d_{A'}(\{l\})$ or $d_{A'}(\{l\}) \geq d_{w'}(H)$. □

Next we give the details about the two actual weights used for the edges in H. We take k as another parameter such that $n \geq k \geq 1$. Let $a = 1 + \frac{1}{2nk}$ and $b = 1 + \frac{1}{k}$, and we note that $a, b \in h^i$ for each edge $h^i \in H$. Furthermore $a < b$. When a deterministic online algorithm requests to update an edge, an uncertainty area is reduced to contain only an actual weight given by the adversary according to Algorithm 9:

Algorithm 9 Adversary for the lower bound of FPVC

```
1: if $n - k$ edges in $H$ have been updated then
2:     Receive $b$;
3: else
4:     Receive $a$
5: end if
```

This brings us to the following two lemmas:

Lemma 9.29. *After $n-1$ updates on $\mathcal{U}$ by any deterministic online algorithm with actual weights obtained as determined by Algorithm 9, the edge uncertainty graph is not solved.*

Proof. Let $\mathcal{U}' = (V, E, A')$ be the edge uncertainty graph after $n-1$ updates have been made on $\mathcal{U}$. In $\mathcal{U}'$ the set H consist of $n-k$ updated edges with trivia uncertainty area $\{a\}$, further $k-1$ updated edges with trivial uncertainty area $\{b\}$, and one more edge that has not been updated and therefore it still has the non-trivial uncertainty area $(1, 3)$. So the uncertainty distance of H in $\mathcal{U}'$ is:
$d_{A'}(H) = (n-k)(1+\frac{1}{2nk}) + (k-1)(1+\frac{1}{k}) + (1\ ,\ 3) = (n + \frac{1}{2k} - k - \frac{1}{2n}) + (k + 1 - 1 - \frac{1}{k}) + (1\ ,\ 3) = (n - \frac{1}{2k} - \frac{1}{2n}) + (1\ ,\ 3) = (n+1-\frac{1}{2k}-\frac{1}{2n}\ ,\ n+3-\frac{1}{2k}-\frac{1}{2n})$
Since $(n+1-\frac{1}{2k}-\frac{1}{2n}\ ,\ n+3-\frac{1}{2k}-\frac{1}{2n}) \not\geq [n+1\ ,\ n+1]$ and
$[n+1\ ,\ n+1] \not\geq (n+1-\frac{1}{2k}-\frac{1}{2n}\ ,\ n+3-\frac{1}{2k}-\frac{1}{2n})$ then, by Lemma 9.28, $\mathcal{U}'$ is not solved. □

Lemma 9.30. *$OPT = k$ for the edge uncertainty graph $\mathcal{U}$ with actual weights obtained as determined by Algorithm 9.*

Proof. We argue that OPT is the number of edges which when updated the uncertainty area becomes $\{b\}$. $\mathcal{U}$ contains k such edges and so let $\mathcal{U}' = (V, E, A')$ be the edge uncertainty graph after updating them in $\mathcal{U}$. $\mathcal{U}'$ further contains $n-k$ edges which have not been updated and therefore each still has the non-trivial uncertainty area $(1, 3)$. So the uncertainty distance of H in $\mathcal{U}'$ is:
$d_{A'}(H) = k(1+\frac{1}{k}) + (n-k)(1\ ,\ 3) = k+1+(n-k\ ,\ 3n-3k) = (n+1\ ,\ 3n-2k+1)$
Since $(n+1\ ,\ 3n-2k+1) > [n+1\ ,\ n+1]$ then, by Lemma 9.28, $\mathcal{U}'$ is solved.

It remains to show that for $k-1$ updates the edge uncertainty graph is not solved, even if the results of the updates were known beforehand. Since $a < b$, for a

number of updates that yield a combination of a's and b's, the uncertainty distance of H cannot be greater than the one of having the same number of updates that yield only b's. So it suffices to show just for the case where $k-1$ updates that yield b's have been made. So let $\mathcal{U}'' = (V, E, A'')$ be the edge uncertainty graph after updating $k-1$ such edges in $\mathcal{U}$. We note $\mathcal{U}''$ further contains $n-k+1$ edges that have not been updated and therefore still have the uncertainty area value $(1, 3)$. So the uncertainty distance of H in $\mathcal{U}''$ is:
$d_{A''}(H) = (k-1)(1+\frac{1}{k}) + (n-k+1)(1\ ,\ 3) = k - \frac{1}{k} + (n-k+1\ ,\ 3n-3k+3) = (n+1-\frac{1}{k}\ ,\ 3n-2k-\frac{1}{k}+3)$
Since $(n+1-\frac{1}{k}\ ,\ 3n-2k-\frac{1}{k}+3) \not\geq [n+1\ ,\ n+1]$ and
$[n+1\ ,\ n+1] \not\geq (n+1-\frac{1}{k}\ ,\ 3n-2k-\frac{1}{k}+3)$ then, by Lemma 9.28, $\mathcal{U}''$ is not solved.

□

Following Lemma 9.29 and Lemma 9.30, there exist configurations of the FPVC problem such that any online strategy can be forced to make n updates to solve the problem whereas, if the results of the updates were known beforehand, given k with $n \geq k$ the problem could have been solved with just k updates. This brings us to the following conclusion for this section:

Theorem 9.31. *There does not exist a deterministic online algorithm for the FPVC problem with constant update competitive ratio.*

We note that our construction could be easily modified to use closed instead of open intervals for the non-trivial uncertainty areas. It remains however open if the problem was to determine all farthest pairs of vertices instead of just one.

Chapter 10

Concluding Remarks and Future Work

10.1 Maximal Points

- For the maximal points verification we have shown in Chapter 5 that, by a reduction from the Minimum Set Cover problem, the problem is NP-hard. In our reduction each uncertainty area is either a singleton or contains two points but it remains open if the same holds when each uncertainty area is connected. Perhaps new results can be found for this case.

- In Chapter 6 we proposed a new model for the maximal point problem and in Section 6.5 of that chapter we presented a polynomial time strategy to solve the verification problem for the new model (MPDPV). In Chapter 7 we have studied the same model but in the setting of coordinate specific updates. In Section 7.3 of that chapter we had a look at the verification problem (MPCSV) and highlighted a few properties that make it harder to construct an algorithm, which are not present in MPDPV. However our results, for the time being, are inconclusive whether a polynomial time strategy is possible. This could be something to further look at in the future.

- In dimensions higher than 2D we have shown in Chapter 8 that there does not exist a deterministic online algorithm with constant competitive ratio. It would be interesting to find new kind of restrictions that allow update-competitive algorithms.

10.2 Convex Hull

- The problem of computing the convex hull of a set of points under uncertainty has been studied in [6]. They present a 3-update competitive algorithm and show that this is the best possible. The result applies only if the input consists of uncertainty areas which are either trivial or closures of connected open areas. Perhaps new results can be found for different restrictions of the problem and/or for its verification setting. Maybe even for dimensionality higher than 2.

10.3 Farthest Pair of Vertices

- We have studied the problem of finding "a" farthest pair of vertices in Chapter 9 for a few different restrictions/types of graphs. It would be interesting to also study the problem where "all" farthest pairs of vertices need to be computable. For example, in Subsection 9.3.2, we have shown that if closed intervals are allowed for uncertainty areas in a tree, and just one farthest pairs of vertices needs to be determined, there does not exist a deterministic online algorithm with constant competitive ratio. This might not be the case if all pairs need to be computable.

10.4 Other Directions

- So far we have seen problems under uncertainty where an update reduces the uncertainty element only on a single input item and has no impact on the uncertainty of others. It would be interesting to find and research a problem in the setting where an update may also influence the uncertainty of other input items. For example lets consider the construction of valid puzzles for the popular number-placement game of sudoku. An instance of the puzzle contains 9×9 cells and is considered valid only if it is uniquely solvable with numbers 1 to 9. So if one wants to create a valid instance, starting possibly from an empty grid, has to place numbers iteratively which not only remove the uncertainty of the current cell but may also reduce the uncertainty of other cells.

- So far we have only studied deterministic algorithms where they behave predictably. The study of randomised algorithms where there is employment of a degree of randomness as part of their logic seems interesting for improvement in the average case.

Bibliography

[1] Jon Louis Bentley, Kenneth L. Clarkson, and David B. Levine. Fast linear expected-time algorithms for computing maxima and convex hulls. *Algorithmica*, 9(2):168–183, 1993.

[2] Kenneth S. Bøgh, Ira Assent, and Matteo Magnani. Efficient gpu-based skyline computation. In *DaMoN*, page 5. ACM, 2013.

[3] Kenneth S. Bøgh, Sean Chester, and Ira Assent. Work-efficient parallel skyline computation for the GPU. *PVLDB*, 8(9):962–973, 2015.

[4] Allan Borodin and Ran El-Yaniv. *Online computation and competitive analysis*. Cambridge University Press, 1998.

[5] Stephan Börzsönyi, Donald Kossmann, and Konrad Stocker. The skyline operator. In *ICDE*, pages 421–430. IEEE Computer Society, 2001.

[6] Richard Bruce, Michael Hoffmann, Danny Krizanc, and Rajeev Raman. Efficient update strategies for geometric computing with uncertainty. *Theory Comput. Syst.*, 38(4):411–423, 2005.

[7] Timothy M. Chan. All-pairs shortest paths with real weights in $O (n ^3/\log n)$ time. *Algorithmica*, 50(2):236–243, 2008.

[8] Timothy M. Chan. More algorithms for all-pairs shortest paths in weighted graphs. *SIAM J. Comput.*, 39(5):2075–2089, 2010.

[9] George Charalambous and Michael Hoffmann. Verification problem of maximal points under uncertainty. In Thierry Lecroq and Laurent Mouchard, editors, *Combinatorial Algorithms - 24th International Workshop, IWOCA 2013, Rouen, France, July 10-12, 2013, Revised Selected Papers*, volume 8288 of *Lecture Notes in Computer Science*, pages 94–105. Springer, 2013.

[10] Liang Chen, Kai Hwang, and Jian Wu. Mapreduce skyline query processing with a new angular partitioning approach. In *IPDPS Workshops*, pages 2262–2270. IEEE Computer Society, 2012.

[11] Fan R. K. Chung. Diameters of graphs: Old problems and new results. *Congressus Numerantium*, 60:295–317, 1987.

[12] Kenneth L. Clarkson. More output-sensitive geometric algorithms (extended abstract). In *FOCS*, pages 695–702. IEEE Computer Society, 1994.

[13] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms (3. ed.)*. MIT Press, 2009.

[14] Wlodzimierz Dobosiewicz. A more efficient algorithm for the min-plus multiplication. *Int. J. Comput. Math.*, 32(1-2):49–60, 1990.

[15] Thomas Erlebach and Michael Hoffmann. Minimum spanning tree verification under uncertainty. In Dieter Kratsch and Ioan Todinca, editors, *Graph-Theoretic Concepts in Computer Science - 40th International Workshop, WG 2014, Nouan-le-Fuzelier, France, June 25-27, 2014. Revised Selected Papers*, volume 8747 of *Lecture Notes in Computer Science*, pages 164–175. Springer, 2014.

[16] Thomas Erlebach and Michael Hoffmann. Query-competitive algorithms for computing with uncertainty. *Bulletin of the EATCS*, 116, 2015.

[17] Thomas Erlebach, Michael Hoffmann, and Frank Kammer. Query-competitive algorithms for cheapest set problems under uncertainty. *Theor. Comput. Sci.*, 613:51–64, 2016.

[18] Tomás Feder, Rajeev Motwani, Liadan O'Callaghan, Chris Olston, and Rina Panigrahy. Computing shortest paths with uncertainty. *J. Algorithms*, 62(1):1–18, 2007.

[19] Tomás Feder, Rajeev Motwani, Rina Panigrahy, Chris Olston, and Jennifer Widom. Computing the median with uncertainty. *SIAM J. Comput.*, 32(2):538–547, 2003.

[20] Robert W. Floyd. Algorithm 97: Shortest path. *Commun. ACM*, 5(6):345, 1962.

[21] Paolo Giulio Franciosa, Carlo Gaibisso, and Maurizio Talamo. An optimal algorithm for the maxima set problem for data in motion. In *Abs. CG'92*, pages 17–21, 1992.

[22] Michael L. Fredman. New bounds on the complexity of the shortest path problem. *SIAM J. Comput.*, 5(1):83–89, 1976.

[23] Harold N. Gabow, Jon Louis Bentley, and Robert Endre Tarjan. Scaling and related techniques for geometry problems. In *STOC*, pages 135–143. ACM, 1984.

[24] Mordecai J. Golin. A provably fast linear-expected-time maxima-finding algorithm. *Algorithmica*, 11(6):501–524, 1994.

[25] Manoj Gupta, Yogish Sabharwal, and Sandeep Sen. The update complexity of selection and related problems. In Supratik Chakraborty and Amit Kumar, editors, *IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science, FSTTCS 2011, December 12-14, 2011, Mumbai, India*, volume 13 of *LIPIcs*, pages 325–338. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2011.

[26] Yijie Han. Improved algorithm for all pairs shortest paths. *Inf. Process. Lett.*, 91(5):245–250, 2004.

[27] Yijie Han. An O (n ^{3}(log log n /log n)$^{5/4}$) time algorithm for all pairs shortest path. *Algorithmica*, 51(4):428–434, 2008.

[28] Yijie Han and Tadao Takaoka. An o(n 3 loglogn/log2 n) time algorithm for all pairs shortest paths. In *SWAT*, volume 7357 of *Lecture Notes in Computer Science*, pages 131–141. Springer, 2012.

[29] Michael Hoffmann, Thomas Erlebach, Danny Krizanc, Matús Mihalák, and Rajeev Raman. Computing minimum spanning trees with uncertainty. In 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*, pages 277–288. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, Germany, 2008.

[30] Katja Hose and Akrivi Vlachou. A survey of skyline processing in highly distributed environments. *VLDB J.*, 21(3):359–384, 2012.

[31] Ju-wook Jang, Madhusudan Nigam, Viktor K. Prasanna, and Sartaj Sahni. Constant time algorithms for computational geometry on the reconfigurable mesh. *IEEE Trans. Parallel Distrib. Syst.*, 8(1):1–12, 1997.

[32] Simon Kahan. A model for data in motion. In Cris Koutsougeras and Jeffrey Scott Vitter, editors, *Proceedings of the 23rd Annual ACM Symposium on Theory of Computing, May 5-8, 1991, New Orleans, Louisiana, USA*, pages 267–277. ACM, 1991.

[33] Sanjiv Kapoor. Dynamic maintenance of maximas of 2-d point sets. In *Proceedings of the tenth annual symposium on Computational geometry*, pages 140–149. ACM, 1994.

[34] Richard M. Karp. Reducibility among combinatorial problems. In Michael Jünger, Thomas M. Liebling, Denis Naddef, George L. Nemhauser, William R. Pulleyblank, Gerhard Reinelt, Giovanni Rinaldi, and Laurence A. Wolsey, editors, *50 Years of Integer Programming 1958-2008 - From the Early Years to the State-of-the-Art*, pages 219–241. Springer, 2010.

[35] S. Khanna and W.-C. Tan. On computing functions with uncertainty. In *Proceedings of the 20th Symposium on Principles of Database Systems (PODS'01)*, pages 171–182, 2001.

[36] David G. Kirkpatrick. Hyperbolic dovetailing. In *ESA*, volume 5757 of *Lecture Notes in Computer Science*, pages 516–527. Springer, 2009.

[37] David G. Kirkpatrick and Raimund Seidel. Output-size sensitive algorithms for finding maximal vectors. In *Symposium on Computational Geometry*, pages 89–96. ACM, 1985.

[38] H. T. Kung, Fabrizio Luccio, and Franco P. Preparata. On finding the maxima of a set of vectors. *J. ACM*, 22(4):469–476, 1975.

[39] Xiaoyong Li, Yijie Wang, Xiaoling Li, Xiaowei Wang, and Jie Yu. GDPS: an efficient approach for skyline queries over distributed uncertain data. *Big Data Research*, 1:23–36, 2014.

[40] Xiaoyong Li, Yijie Wang, and Jie Yu. An efficient scheme for probabilistic skyline queries over distributed uncertain data. *Telecommunication Systems*, 60(2):225–237, 2015.

[41] Maarten Löffler and Marc J. van Kreveld. Largest and smallest convex hulls for imprecise points. *Algorithmica*, 56(2):235–269, 2010.

[42] Nicole Megow, Julie Meißner, and Martin Skutella. Randomization helps computing a minimum spanning tree under uncertainty. In *ESA*, volume 9294 of *Lecture Notes in Computer Science*, pages 878–890. Springer, 2015.

[43] Kasper Mullesgaard, Jens Laurits Pederseny, Hua Lu, and Yongluan Zhou. Efficient skyline computation in mapreduce. In *EDBT*, pages 37–48. OpenProceedings.org, 2014.

[44] Chris Olston and Jennifer Widom. Offering a precision-performance tradeoff for aggregation queries over replicated data. In Amr El Abbadi, Michael L. Brodie, Sharma Chakravarthy, Umeshwar Dayal, Nabil Kamel, Gunter Schlageter, and Kyu-Young Whang, editors, *VLDB 2000, Proceedings of 26th International Conference on Very Large Data Bases, September 10-14, 2000, Cairo, Egypt*, pages 144–155. Morgan Kaufmann, 2000.

[45] Seth Pettie and Vijaya Ramachandran. A shortest path algorithm for real-weighted undirected graphs. *SIAM J. Comput.*, 34(6):1398–1431, 2005.

[46] Franco P. Preparata and Michael Ian Shamos. *Computational Geometry - An Introduction*. Texts and Monographs in Computer Science. Springer, 1985.

[47] Tadao Takaoka. A new upper bound on the complexity of the all pairs shortest path problem. *Inf. Process. Lett.*, 43(4):195–199, 1992.

[48] Tadao Takaoka. A faster algorithm for the all-pairs shortest path problem and its application. In *COCOON*, volume 3106 of *Lecture Notes in Computer Science*, pages 278–289. Springer, 2004.

[49] Tadao Takaoka. An o(n^3loglog$n$/log$n$) time algorithm for the all-pairs shortest path problem. *Inf. Process. Lett.*, 96(5):155–161, 2005.

[50] Kuan-Chieh Robert Tseng and David G. Kirkpatrick. Input-thrifty extrema testing. In Takao Asano, Shin-Ichi Nakano, Yoshio Okamoto, and Osamu Watanabe, editors, *Algorithms and Computation - 22nd International Symposium, ISAAC 2011, Yokohama, Japan, December 5-8, 2011. Proceedings*, volume 7074 of *Lecture Notes in Computer Science*, pages 554–563. Springer, 2011.

[51] Stephen Warshall. A theorem on boolean matrices. *J. ACM*, 9(1):11–12, 1962.

[52] Ryan Williams. Faster all-pairs shortest paths via circuit complexity. In *STOC*, pages 664–673. ACM, 2014.

[53] Boliang Zhang, Shuigeng Zhou, and Jihong Guan. Adapting skyline computation to the mapreduce framework: Algorithms and experiments. In *DASFAA Workshops*, volume 6637 of *Lecture Notes in Computer Science*, pages 403–414. Springer, 2011.

[54] Xu Zhou, Kenli Li, Yantao Zhou, and Keqin Li. Adaptive processing for distributed skyline queries over uncertain data. *IEEE Trans. Knowl. Data Eng.*, 28(2):371–384, 2016.

[55] Uri Zwick. Exact and approximate distances in graphs - A survey. In *ESA*, volume 2161 of *Lecture Notes in Computer Science*, pages 33–48. Springer, 2001.

[56] Uri Zwick. A slightly improved sub-cubic algorithm for the all pairs shortest paths problem with real edge lengths. In *ISAAC*, volume 3341 of *Lecture Notes in Computer Science*, pages 921–932. Springer, 2004.