SOME RESULTS IN UNIVERSAL AND A PRIORI OPTIMIZATION

A Dissertation

Presented to the Faculty of the Graduate School

of Cornell University

in Partial Fulfillment of the Requirements for the Degree of

Doctor of Philosophy

by

Frans Schalekamp

May 2007

SOME RESULTS IN UNIVERSAL AND A PRIORI OPTIMIZATION

Frans Schalekamp, Ph.D.

Cornell University 2007

In this thesis, some combinatorial problems are studied in the light of the a priori and universal optimization framework, as introduced for the Traveling Salesman Problem by Jaillet, and by Bartholdi & Platzman, respectively. In both the a priori, as well as the universal framework, there is an underlying optimization problem, and (advance) knowledge of the potential instances that can arise. The goal is to find a "master solution" of a special form, that in turn fully specifies the solution to any potential instance of the optimization problem that arises. In the case of the traveling salesman problem, the master solution is a tour on the complete set of all customers in the potential instances. This master tour now specifies the solution for each instance as follows: the customers in the instance will be visited in the same order as the master solution.

The results of this thesis include an optimization algorithm for the a priori and universal Traveling Salesman Problem on tree metrics, an $O(\log n)$-approximation algorithm for the a priori Traveling Salesman Problem, with no restrictions on the metric space and no knowledge about the probability distribution on the potential instances, an approximation algorithm for a priori and universal $1 \mid pmtn, r_j \mid \sum C_j$, and a nearly optimal algorithm for AdWords.

We also study the real life problem of creating Nearly Isogenic Lines (NILs). The solution that we propose can be seen as an a priori solution, since the solu-

tion that we propose is nonadaptive. Finally, in addition to dealing with a priori and universal optimization, we also give a Fully Polynomial Time Approximation Scheme (FPTAS) for a certain Stochastic Dynamic Programs, with an application in option pricing.

BIOGRAPHICAL SKETCH

Frans Schalekamp was born in 1974, in Naarden, the Netherlands. In the 1980s Frans broke his arm during ice skating. To keep him quiet, his father bought a computer. This resulted in many hours of playing computer games (by today's standards rather tame ones), as well as the first introduction to algorithms. Imagine his surprise when he learned that his $O(n^2)$ sorting algorithm could dramatically be improved.

After a fun time in high school in Wassenaar, Frans discovered that there actually exists a course of study that focuses on studying algorithms: Operations Research (then plainly called "Econometrie"). He chose to move to the lively capital of the Netherlands to persue his destiny at the Vrije University. Upon completion of his undergraduate degree, he worked for a brief period as a developer for ORTEC Systems in Gouda, after which he moved to the United States, more in particular to Ithaca, where he just now finished his PhD in Operations Research. This last year he has also been employed by Nature Source Genetics, where he applies his knowledge of optimization to the wondrous world of biology.

ACKNOWLEDGEMENTS

There are a great many people who have had a very positive influence on my life, and I would like to thank all of you. I hope to have the opportunity to be a positive force in somebody else's life like you have been in mine.

Now, running the risk of insulting the people that I do not mention by name, I want to mention some people in particular. This list is not by any means meant to be complete.

First of all I would like to thank David Shmoys, not only for being my advisor, but also for helping me and Anke, and making our life easier and better in Ithaca in many ways. I would also like to thank the other members of my committee, Sid Resnick and Jon Kleinberg.

Then, I would like to mention some of the people that I met here, and that made my life in Ithaca a lot of fun: Tuncay Alparslan, Gavin Hurley (congratulations with your wedding next year), Millie Chu (who has caught up on tv), Sam Ehrlichman, Emmanuel Sharef, Soumyadip Ghosh (congratulations with your wedding this month), Alex Kahn (and Jenny, congratulations with Ben), Stefan Wild (and Hilary), Sam Steckley, Bharath Rangarajan (who learned to enjoy Belgian beers, but only after he left Ithaca), Kazu Shimbo, Davina Kunvipusilkul, Vardges Melkonian, Amar Sapra, Tim Huh, Yun Shi and Chen Li. Thanks for all the good times.

Even though during my time here I did not speak to Wolfert Brederode as often as I would like, let alone go to his concerts, I consider him to be a great friend (and a great musician).

From my brief period at ORTEC, I would like to thank Joaquim Gromicho. He taught me the joy of patterns, and is great guy in general.

My time at the Vrije University would not have been the same without Professors Henk Tijms and Rein Nobel. I thank them both for their encouragement and advice to pursue a PhD.

Looking back even further, I would also like to thank some of the teachers at the Rijnlands Lyceum in Wassenaar, my high school, who went out of their way to try and interest me in science.

Of course I would like to thank my family and extended family for their encouragement and bearing with me these many years of schooling.

Finally I would like to thank Anke for putting up with me all these years. I know that I would not be what I am today if it were not for her.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

PREFACE

In this thesis, some combinatorial problems will be studied in the light of the *a priori* and universal optimization framework, as introduced for the Traveling Salesman Problem by Jaillet [21], and by Bartholdi & Platzman [2], respectively. In both the *a priori*, as the well as the universal framework, there is an underlying optimization problem, and (advance) knowledge of the potential instances that can arise. The goal is to find a "master solution" of a special form, that in turn fully specifies the solution to any potential instance of the optimization problem that arises. In the case of the traveling salesman problem, the master solution is a tour on the complete set of all customers in the potential instances. This master tour now specifies the solution for each instance as follows: the customers in the instance will be visited in the same order as the master solution.

In the case of universal optimization, one is interested in the *worst case* ratio of the solution obtained from the master solution for an instance and the optimal solution to the instance, where the worst case is with respect to any potential instance.

In the *a priori* framework, probabilities are associated with each of the possible instances (often in an indirect way, because of complexity issues that may arise from specifying a probability distribution — we will ignore this). The goal now is to get a good *average case* solution; the *expected value* of the objective is to be optimized.

The results of this thesis include an optimization algorithm for the *a priori* and universal Traveling Salesman Problem on tree metrics, an $O(\log n)$-approximation algorithm for the *a priori* Traveling Salesman Problem, with no restrictions on the metric space and no knowledge about the probability distribution on the potential

instances, an approximation algorithm for *a priori* and universal $1 \mid pmtn, r_j \mid \sum C_j$, and a nearly optimal algorithm for AdWords.

In addition to dealing with *a priori* and universal Optimization, this thesis has two more chapters. The last chapter of this thesis, on Approximate Dynamic Programming, is again rather theoretical. It is unrelated to the other chapters in the sense that this chapter has no connections with *a priori* or universal optimization.

Finally, the solution that we propose to the real life problem of creating nearly isogenic lines in Chapter 4, can be seen as an *a priori* solution, since the solution that we propose is nonadaptive. The flavor of this chapter is completely different than that of all the other chapters, in the sense that in this chapter the focus is on getting a practical solution to a real life problem, more than on obtaining theoretical results. Understanding the problem and formulating it in a way that was true to the problem at hand, was actually one of the more difficult challenges that we faced for this chapter.

Chapter 1

A Priori and Universal Traveling Salesman Problem

1.1 Introduction

Consider the following problem: A delivery person needs to serve clients each day. Her (potential) client set is fixed and known, but each day only a subset of the potential client set needs to be served. Instead of reoptimizing every day, the delivery person would like to have a tour of the potential client set, such that if she travels the subtour induced from the complete tour for that day's subset of clients (i.e., she visits the clients in the same order as the complete tour, shortcutting the tour where clients are absent), then she does not travel much more than had she used the optimal tour for this particular subset of clients.

More precisely, two different problems can be considered. One is an *adversial* model, where any of the subsets of clients is possible, and the objective is to minimize the maximum ratio, over all possible client subsets, of the length of the induced subtour to the length of the optimal tour for that subset. (Another way to view this is that the optimizer announces the complete tour, and next the adversary can decide which subset of clients actually need service). This is known as the *universal traveling salesman problem.*

A second model is a *probabilistic* model. Here one assumes that there is some probability distribution on subsets of clients and the objective is to minimize the expected length of the induced subtour. This problem is known as the *a priori traveling salesman problem.* (Note that there are complexity issues arising from

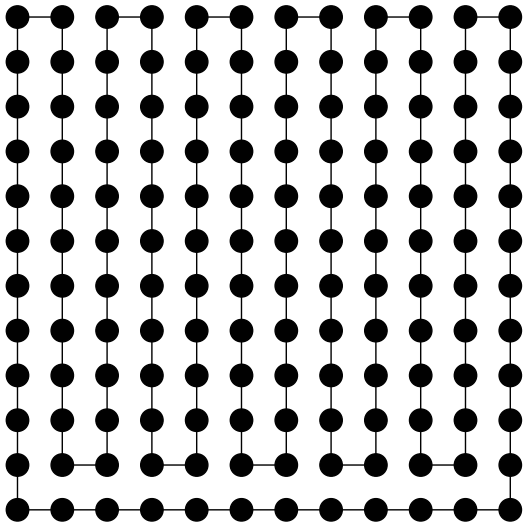


Figure 1.1: Bad ($\Omega(\sqrt{n})$) example for straightforward approach of just using the traveling salesman tour on the full set of clients. The edges that are drawn have length $1 - \varepsilon$, the edges that are not drawn have distance equal to the Euclidean distance, where the distance between two nodes that are closest is equal to 1.

specifying the probability distribution.)

The example in Figure 1.1 shows that the straightforward approach of just using the traveling salesman tour on the full set of clients (assume for sake of argument that we can do this, even though the problem is NP-hard) can be quite bad. The edges that are drawn in Figure 1.1 have length $1 - \varepsilon$, the edges that are not drawn have distance equal to the Euclidean distance, where the distance between two nodes that are closest is equal to 1. More precisely, the example shows that this approach is as bad as $\Omega(\sqrt{n})$. (Proof: First of all note that the edges that are drawn in Figure 1.1 form the TSP on the full set of clients: every node has to connect with some other node, and the minimum traveling distance between two clients is equal to $1 - \varepsilon$. The tour comprised of the edges drawn attains this lower bound and is therefore optimal. Now, consider the following

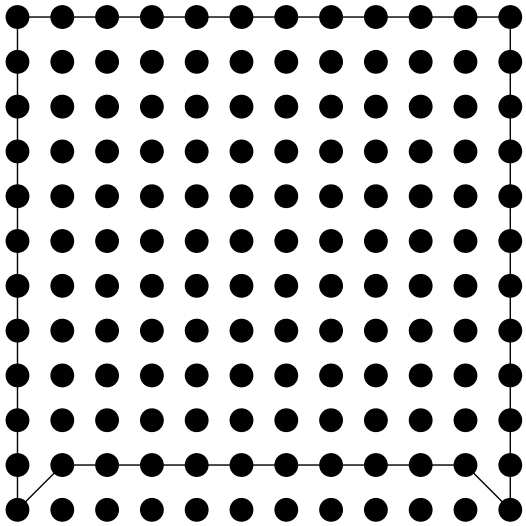


Figure 1.2: Feasible tour of length $O(\sqrt{n})$ for this particular scenario.

subset of clients: assume that only the set of the clients need service, for which the TSP tour changes direction. The shortcutted tour will be the same as the original tour, with length $n(1-\varepsilon)$, whereas a feasible tour of length $O(\sqrt{n})$ is depicted in Figure 1.2. This proves the gap for the Universal case. For the A Priori case we further need to specify a probability distribution: assuming that the "bad" scenario has probability $O(1)$ does the trick.)

1.1.1 Our Results

In this chapter we will show that when the client distances form a *tree metric*, we can, quite surprisingly, compute a universally optimal tour, solving both problems optimally, and we can even solve the probabilistic model without any knowledge of the probability distribution.

This special case of tree metrics is particularly significant, because any metric can be (probabilistically) embedded in tree metrics with expected distortion $O(\log n)$ (where n is the number of points in the metric), as shown by Fakcharoen-

phol, Rao and Talwar [15] (building on Bartal's work [1]) — we will explain this method in section 1.2.1. As a corollary we therefore get that the *a priori* TSP on *any* metric space can be solved within $O(\log n)$ of optimal, even without any knowledge of the distribution on the subsets. It is interesting to note that this is the same guarantee as Platzman and Bartholdi's result for the universal TSP in the Euclidean plane [2, 26].

1.1.2 Relevant Literature

The universal TSP was motivated by getting good solutions for the "Meals On Wheels" program of Senior Citizen Services Inc., "which delivers prepared lunches to people who are unable to shop or cook for themselves"[3]. The list of clients that need service in this particular application is quite volatile "...because of the nature of the clients: most are elderly or ill. They may die, or recover from illness, or receive care elsewhere..."[3]. The approximation algorithm of [2, 26] uses a spacefilling curve to map points from the plane to the unit line (interpreted as a circle), on which a traveling salesman tour is trivial to solve. They proved a performance guarantee of $O(\log n)$, and conjectured that it was really $O(1)$. Bertsimas and Grigni [6] disproved this conjecture for Bartholdi and Platzman's algorithm, exhibiting a (family of) counterexample(s) for which $O(\log n)$ is tight up to a multiplicative constant.

Recently the universal TSP has again become an object of study, with Jia et al. [23] giving an algorithm for which the induced tour for any subset is within an $O(\log^4 n/\log\log n)$ factor of optimal, and even more recently Gupta, Hajiaghayi and Räcke [17] improve this bound to $O(\log^2 n)$, using the ideas of Fakcharoenphol, Rao and Talwar [15]. A lower bound for the 2-dimensional Euclidean universal

TSP (i.e., independent of an algorithm) of $\Omega(\sqrt[6]{\log n / \log\log n})$ was presented by Hajiaghayi, Kleinberg and Leighton [18].

Jaillet introduced the notion of *a priori* optimization, in particular the *a priori* TSP [21, 22]. Except for asymptotic results on points distributed independently and uniformly on the plane [22, 7, 5], not much is known. (Note, of course, that any guarantee for the universal TSP implies the same guarantee for the *a priori* TSP.)

In [13] and [11] a related question was studied, namely, for what metrics does there exist a universal TSP solution such that the induced subtours are actually optimal solutions to the smaller problems? Or phrased differently, what metrics give a universal TSP solution with optimal ratio equal to 1? This property is called the master tour property and it turns out that the class of metrics that possess this property is fully described and known as *Kalmanson metrics*, as shown in [13].

1.2 A Priori and Universal TSP

We start by introducing some notation needed to present our results. For the traveling salesman problem, the the input consists of a finite set of points $V = \{1, 2, \ldots, n\}$ and a "distance" function $d : V \times V \to Q^{\geq 0}$; we wish to find a tour τ, given by a permutation of all points in V, such that the length of the tour, $d(\tau) = \sum_{i=1}^{n-1} d(\tau_i, \tau_{i+1}) + d(\tau_n, \tau_1)$ is minimized.

For the *a priori* and universal TSP, only a subset $S \subseteq V$ will be "active". For a tour τ of V, we define $\tau(S)$ to be the tour restricted to S, i.e. $(\tau(S))_i = \tau_j$ where j is such that $\tau_j \in S$ and $\#\{\tau_k \in S, k \leq j\} = i$ (where $\#\{N\}$ denotes the cardinality of the set N). For each $S \subseteq V$, we also let $OPT(S)$ denote the optimal tour on S and let $d(OPT(S))$ denote its length.

The input and output for the *universal TSP* are the same as for the TSP: we are given V and d, and we want a tour τ. The objective, however, now is to minimize, over all subsets $S \subseteq V$, the ratio of the length of the tour induced by the permutation τ on S, divided by the length of the optimal tour on S, i.e. to minimize $\max_{S \subseteq V}[d(\tau(S))/d(OPT(S))]$.

For the *a priori TSP*, the input is again a set of points V, and a distance function d, but there is also a probability distribution specified over the subsets of V. For our results, we do not need any knowledge of this distribution. The goal now is to find a tour τ that minimizes the *expected length* (with respect to the probability distribution on subsets of V) of the tour induced by τ; i.e., to minimize $E_S[d(\tau|S)]$.

Let V be a finite set of points. A distance function $d : V \times V \to Q^{\geq 0}$ is called a *metric* if (1) $d(i,i) = 0$ for all $i \in V$ (2) $d(i,j) = d(j,i)$ for all $i,j \in V$, and (3) $d(i,j) \leq d(i,k) + d(k,j)$ for all $i,j,k \in V$ (the triangle inequality). (To be precise, this is the definition of a semimetric. For a semimetric to be a metric we also want $d(i,j) > 0$ if $i \neq j$, but this is of no importance for this chapter.) Furthermore, we say that d is a *tree metric* if there exists a tree $T = (V_T, E_T)$ with nodes $V_T \supseteq V$ and lengths associated with the edges, such that $d(i,j)$ is exactly equal to the length of the unique path in T from i to j, for all $i,j \in V$. For each subset $S \subseteq V$, one can obtain an *induced tree* $T(S)$, which is the union over all pairs of nodes $i,j \in S$ of the edges (and nodes) in the path between i and j. We can now state and prove our results.

Lemma 1. *Consider an input to the TSP given by a tree metric on V that is realized by the tree T; then for any subset $S \subseteq V$, the length of any tour on S is at least twice the total length of the edges in the induced tree $T(S)$.*

Proof. Take any traveling salesman tour τ on S. We can transform this to a walk using the edges of $T(S)$, by inserting, between each pair of consecutive points i and j in τ, the unique path in T connecting i and j; this does not change the length of the tour. So we can view any tour as a walk that only uses the edges of the tree $T(S)$ (multiple times). However, consider any edge in $T(S)$; if we delete this edge, then we separate the tree $T(S)$ into two components, thereby partitioning S into two non-empty subsets S_1 and S_2. Thus, the walk must contain this edge at least twice in the tour, at least once "entering" S_1 and at least once "leaving" it. □

For any tree T, we can consider the walk formed by traversing "around the outside" of T: we shall call this walk $\mathsf{trav}(T)$. Such a walk can be shortcut in a number of ways to obtain a tour; for a tree metric, all such tours are the same length, and by Lemma 1 are all optimal solutions to the TSP. It is easy to see that if we start with one such tour τ, and consider the tour $\tau(S)$ for any subset $S \subseteq V$, then this tour can be obtained by shortcutting the walk $\mathsf{trav}(T(S))$. Hence $\tau(S)$ is an optimal solution to the TSP input induced by S, and we have shown the following result.

Theorem 2. *For any tree metric, any shortcutting of the walk* $\mathsf{trav}(T)$ *yields an optimal solution to the universal TSP, and therefore an optimal solution to the a priori TSP.*

Note that the results above are also implied by the fact that tree metrics are a subclass of Kalmanson metrics, and the fact that Kalmanson metrics have the master tour property as proved in [13]. The proofs here are much simpler, as we just consider tree metrics.

1.2.1 Aside: The FRT Procedure

For completeness, we will now briefly describe the FRT procedure, due to and named after Fakcharoenphol, Rao and Talwar [15], building on Bartal's work [1].

Their main result (Theorem 2 in [15]) is the following: given any metric d, there is exists a probability distribution on tree metrics, say $\{d_T\}_T$, such that $d^T(i,j) \geq d(i,j)$ for all $i,j \in V$ and for all T, and $E_T[d^T(i,j)] = O(\log(n))d(i,j)$ for all $i,j \in V$.

Moreover, we can easily sample from this distribution, as follows: A tree metric is made by repeated refining of partitions of V, which are then interpreted as a tree, in which each of the original nodes are leaves, and sets in the partitions are intermediate nodes. The correct choice of weights on the edges now gives the result. We will now make the ingredients of the algorithm more precise.

The main subprocedure needed is a procedure that partitions sets.

Subprocedure PARTITION

IN: Metric d on V, set $S \subset V$, permutation π of the nodes in V, a number x

OUT: Partitioning of S into $S'_1, S'_2, \ldots, S'_k$ for some k

1. **for** $k \leftarrow 1$ **to** $|V|$ **do**

   - $S'_k \leftarrow (B(\pi_k, x) \cap S) \backslash \left(\bigcup_{i=1}^{k-1} S'_i \right)$

     where $B(v,x) = \{u \in V : d(u,v) \leq x\}$, the ball of radius x around v

2. **return** $\{S'_1, S'_2, \ldots, S'_k\} \backslash \{\emptyset\}$.

Let, for ease of exposition, the smallest distance between any two points be at least 1, and the maximum distance exactly 2^δ (this is straightforwardly done using scaling).

The main procedure now is as follows:

1. Choose a random permutation π of the nodes in V.
2. Choose β in $[1, 2]$ randomly from the distribution $p(x) = 1/(\ln 2x)$.
3. Let $P_{\delta-1}$ be the partition of V that just consists of V.
4. **for** $i \leftarrow \delta - 2$ **downto** -1 **do**
5. for each set S in P_{i+1}, call PARTITION with parameters d, S, π and $2^i\beta$, thus creating a new partition P_i

The tree metric that is created has nodes for every set in every partition, and egdes between sets only if they belong to partitions in P_i and P_{i-1} for some i, and one is a subset of the other (they may be the same set). The weight on the edge is 2^i.

For a proof that this procedure will yield the desired result, we refer the reader to [15]. Figures 1.3, 1.4, 1.5, 1.6 and 1.7 show an example of the FRT procedure in action on a Euclidean instance. The nodes are numbered according to the order of the random permutation.

1.2.2 Using the FRT Procedure

We will now show how we can use the FRT procudure.

Corollary 3. *There is a polynomial-time randomized algorithm that, for any metric input d to the TSP on n points, computes a tour τ such that for each subset $S \subseteq V$, the expected cost of the tour $\tau(S)$ is $O(\log n)d(OPT(S))$ (where the expectation is with respect to the random choices of the algorithm).*

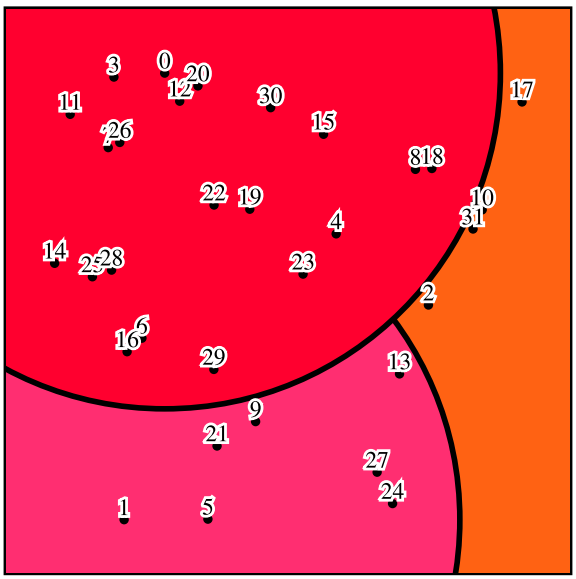


Figure 1.3: Step 1 in the FRT algorithm

Proof. Use (Bartal's) stochastic tree embedding [1, 14, 15]: this gives a distribution over tree metrics, d^T, such that $d^T(i,j) \geq d(i,j)$ for all $i,j \in V$, and $E_T[d^T(i,j)] = O(\log(n))d(i,j)$ for all $i,j \in V$. (We will use E_T to denote the expectation with respect to this distribution over the tree metrics.) For each subset $S \subseteq V$ and each T in the distribution, note that $d^T(\mathsf{trav}(T(S)) \leq d^T(OPT(S))$, since $OPT(S)$ is the optimal tour for S with respect to the original distance metric d, whereas the traversal yields the optimal tour for d^T. Therefore,

$$E_T[d^T(\mathsf{trav}(T(S)))] \leq E_T[d^T(OPT(S))]$$

for each $S \subseteq V$. We can now conclude that

$$E_T[d(\mathsf{trav}(T(S)))] \leq E_T[d^T(\mathsf{trav}(T(S)))]$$

$$\leq E_T[d^T(OPT(S))] = O(\log n)d(OPT(S)),$$

where the first and third inequalities are due the properties of the embedding (non-shrinking, and in expectation expanding by at most $O(\log n)$). ∎

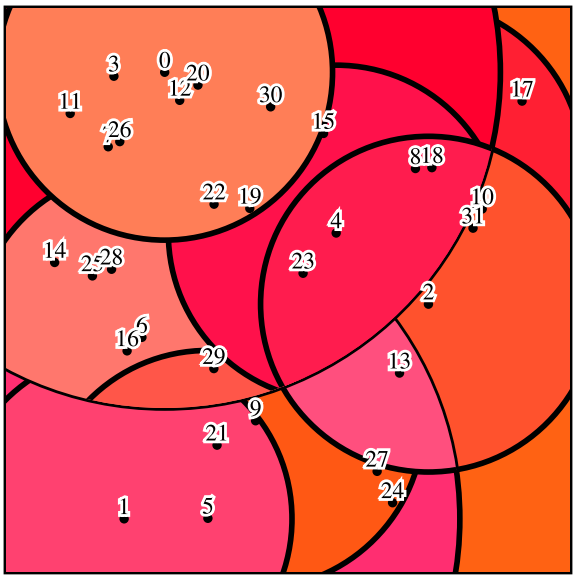


Figure 1.4: Step 2 in the FRT algorithm

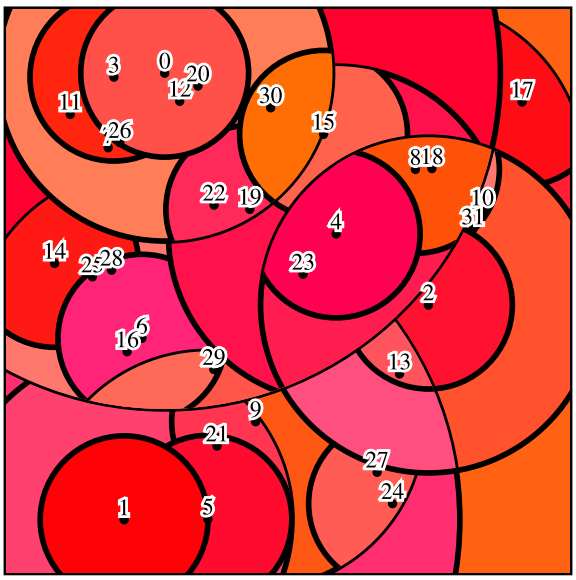


Figure 1.5: Step 3 in the FRT algorithm

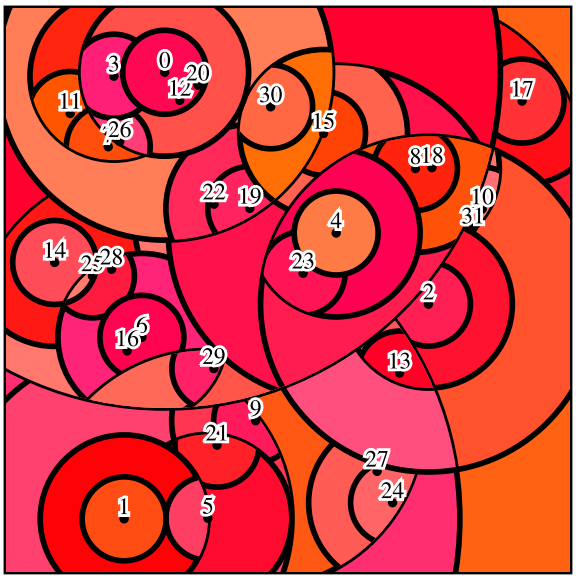


Figure 1.6: Step 4 in the FRT algorithm

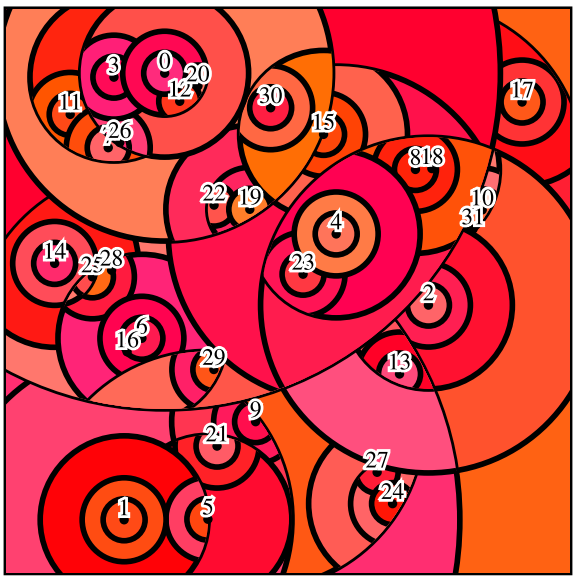


Figure 1.7: Step 5 in the FRT algorithm

Corollary 4. *There is a randomized $O(\log n)$-approximation algorithm for the a priori TSP on any metric space with n points — even without specifying the probability distribution on the subsets.*

Proof. The result follows rather directly from Corollary 3. Denote by p_S the probability that subset S "occurs". The expected cost (with respect to the random tree metric) of our *a priori* TSP solution (which involves an expectation with respect to the choice of S) is

$$\begin{aligned} E_T[\sum_S p_S d(\mathsf{trav}(T(S)))] &= \sum_S p_S E_T[d(\mathsf{trav}(T(S)))] \\ &= \sum_S p_S O(\log n) d(OPT(S)) \\ &= O(\log n) \sum_S p_S d(OPT(S)) \\ &= O(\log n) E_S[d(OPT(S))]. \end{aligned}$$

If $APOPT(S)$ denotes the optimal *a priori* tour restricted to S, then $d(OPT(S)) \leq d(APOPT(S))$ for each S, and so $E_S(d(OPT(S))) \leq E_S(d(APOPT(S)))$, which is exactly the optimal value for the *a priori* TSP. □

Note however, that this does *not* imply a $O(\log n)$ bound for the universal TSP problem. The universal TSP is concerned with $\max_S d(\mathsf{trav}(T(S))/d(OPT(S))$, and we cannot conclude anything directly about this quantity based on the arguments above.

The fact that the adversary can choose the particular subset *after* seeing the tour resulting from the FRT procedure (i.e., the adversary is *adaptive*) is important here. If we weaken the adversary, and make her *non-adaptive* (i.e., she has to choose the subset *before* the FRT procedure is run), then we do have an $O(\log n)$ algorithm, as implied by the result by Ben-David, Borodin, Karp, Tardos and Wigderson [4] (see also [8]).

It is interesting to note that we can obtain analogous results for variants of the *a priori* and universal minimum spanning tree and Steiner tree problems as introduced by Bertsimas [5].

Finally, note that the result in [24] implies that it is not possible to find a better approximation for the *a priori* TSP using the technique of probabilistically embedding the metric in *cut metrics* (a cut metric is a metric d that can be represented as $d(i,j) = \sum_{S:|\{i,j\}\cap S|=1} w_S$), of which tree metrics are a subclass.

1.3 Alternative Proofs

Let d be a tree metric and $T = (V, E)$ be the tree associated with the metric. Define $\{y_S\}_S$ as follows: for every node v in V except the root, let $T_v = (S_v, E_v)$ be the subtree with v as the root, that includes all descendents of v, and let y_{S_v} be the length of the edge adjacent to v on the path to the root. Let $y_S = 0$ for all other $S \subseteq V$.

Since the path between nodes u and v in a tree can be decomposed in the path from u to the least common ancestor of u and v, and the path from this node to v, it readily follows that we can express the tree metric in the following way.

Observation 5. $d(i,j) = \sum_{i\in S, j\notin S} y_S$.

Further, we also know that the following linear program gives a lower bound on the length of the optimal TSP (known as the Held-Karp lower bound) for metrics:

$$\begin{aligned} \text{minimize } & \sum_{ij} d_{ij} x_{ij} \\ \text{s.t. } & \sum_{i \in S, j \notin S} x_{ij} \geq 2 \qquad \forall S \subset V, S \neq \emptyset, S \neq V \\ & x_{ij} \geq 0. \end{aligned}$$

The dual of this LP is

$$\begin{aligned} \text{maximize } & 2 \sum_{S} y_S \\ \text{s.t. } & \sum_{S: i \in S, j \notin S} y_S \leq d_{ij} \qquad \forall i, j \in V \\ & y_S \geq 0. \end{aligned}$$

Lemma 6. *Let $\bar{x}_{ij} = 2$ if ij is an edge in the tree associated with the tree metric (this exactly corresponds to the TSP tour without shortcutting). Then $\bar{x}$ is an optimal primal solution.*

Proof. First we will show that $\bar{x}$ is a feasible solution. Take any $S \subset V$, so that $S \neq \emptyset, S \neq V$. Because of symmetry, we can assume that S does not contain the root (otherwise replace S by $V \backslash S$, this will give the same constraint). For all nodes in S, there is an edge adjacent to it that is on the path to the root. Since the root is not in S, at least one of these edges is such that the other endpoint is not in S. Therefore $\bar{x}$ is feasible.

We will prove optimality by showing that y_S is a complementary feasible dual solution. The feasibility of y_S follows directly from Observation 5 and the fact that y_S are defined in terms of distances, and hence are nonnegative. Note that $y_S > 0$ implies that S is the node set of a subtree that includes all descendents of some

node i. Note that x_{ij} is equal to 2 for j which is the direct parent of i. Further, x_{ij} equals 0 for all other $j \notin S$, since there is only one edge going towards the root for each node. So the dual complementary slackness conditions hold.

Conversely, suppose $x_{ij} > 0$. This implies that ij is an edge of the tree, and therefore $y_S = d_{ij}$, where S is the node set of the full subtree of either i or j (whichever is the child of the other). Moreover, this is the only such subtree that includes only one of i and j, and therefore $y_S = 0$ for all other S such that $|S \cap \{i, j\}| = 1$. So the primal complementary slackness conditions hold, and we can conclude that $\bar{x}$ is an optimal primal solution. □

We can now reprove Theorem 2 in this slightly different format (note that here $x_{ij} = 2$ is replaced by $x_{ij} = 1$ and $x_{ji} = 1$).

Proposition 7. *Let (V, d) be a metric space. Suppose there exists a TSP tour on (V, d), that meets the Held-Karp lower bound where all dual constraints are met with equality for the optimal dual solution. Then this is a universal TSP tour, where the tour restricted to any subset V' of V is actually* optimal.

Proof. Let $\bar{x}$ be an optimal binary primal solution, and $\bar{y}$ be an optimal dual solution. Since there exists a TSP tour on that meets the Held-Karp lower bound, we know that complementary slackness conditions hold.

Complementary slackness gives us that $\bar{y}_S > 0$ implies $\sum_{i \in S, j \notin S} \bar{x}_{ij} = 2$ (and that $\bar{x}_{ij} = 1$ implies $\sum_{S: i \in S, j \notin S} \bar{y}_S = d_{ij}$).

Now look at $V' \subseteq V$. Define x' is the tour induced by $\bar{x}$ restricted to V', i.e.

$$x'_{ij} = \begin{cases} 1 & \text{if there is an } i \text{ to } j \text{ path in } \bar{x} \text{ that only uses intermediary nodes} \\ & \text{that are in } V \backslash V' \\ 0 & \text{otherwise} \end{cases}$$

for $i, j \in V'$, and

$$y'_{S'} = \sum_{S \subseteq V : S \cap V' = S'} \bar{y}_S$$

for $S' \subseteq V'$.

We want to show now that x' and y' are feasible solutions for which the complementary slackness conditions hold, which would prove that the solutions are optimal.

First of all, x' is obviously a tour on V'. Further, y' is dual feasible:

$$\begin{aligned}
\sum_{S' \subseteq V' : i \in S', j \in V' \setminus S'} y'_{S'} &= \sum_{S' \subseteq V' : i \in S', j \in V' \setminus S'} \sum_{S \subseteq V : S \cap V' = S'} \bar{y}_S \\
&= \sum_{S \subseteq V : i \in S, j \in V \setminus S} \bar{y}_S \\
&= d_{ij}
\end{aligned}$$

for all i, j, where the last equality follows from the fact that all dual inequalties are met with equality for $\bar{y}$.

Since all dual inequalities hold with equality, we only have to check dual complementary slackness conditions. Suppose $y'_{S'} > 0$. By definition of $y'_{S'}$ and nonnegativity of $\bar{y}$, we get that there exists $S \subseteq V$ such that $S \cap V' = S'$ and $y_S > 0$. Thus, $\sum_{i \in S, j \in V \setminus S} \bar{x}_{ij} = 2$.

Let j_1 and $j_2 \in V \setminus S$ be such that $x_{i_1 j_1} = x_{i_2 j_2} = 1$. If j_1 and/or $j_2 \notin V'$ follow the original tour to the first node in V' — call these k_1 and k_2 (so these may be equal to j_1 and j_2). Similarly, if i_1 and/or $i_2 \notin V'$ follow the original tour to the first node in V' — call these ℓ_1 and ℓ_2. We now know there is a path from ℓ_1 (ℓ_2) to k_1 (k_2), which only uses intermediary nodes that are in $V \setminus V'$, so $x'_{\ell_1 k_1} = x'_{\ell_2 k_2} = 1$. So $\sum_{i \in S', j \in V' \setminus S'} \bar{x}_{ij} \geq 2$.

Now suppose, $\sum_{i \in S', j \in V' \setminus S'} \bar{x}_{ij} > 2$, i.e. $x_{\ell_3 k_3} = 1$ for some $\ell_3 \notin \{\ell_1, \ell_2\}$. That means there exists some third path in the original tour that has to leave S, but there are only two since $\sum_{i \in S, j \in V \setminus S} \bar{x}_{ij} = 2$. $\square$

Chapter 2

A Priori and Universal $1 \,|\, \mathrm{pmtn}, r_j \,|\, \sum C_j$

2.1 Introduction

One of the big areas within Operations Research is concerned with scheduling. We will now examine one scheduling problem in the light of the *a priori* and universal Optimization framework, as introduced by Jaillet [21, 22].

Similarly to the *a priori* and universal TSP, we will assume that we have full knowledge of the set of potential jobs. The jobs that actually have to be processed are revealed to us after we have settled on an overall schedule, that somehow defines a schedule for any job subset as well. We choose the following approach: the overall schedule will be a permutation of all potential jobs, and the schedule for a particular subset of jobs will be defined by this permutation by interpreting this permutation as a preference, i.e., at every point in time the first job available of the subset is processed.

Jobs in the problem we study have *processing times*, the time that that particular job has to spend on the machine, and release dates, the time after which the job becomes available (in that the job cannot be processed on the machine before this time). Further, we assume that *preemption* is allowed, i.e., the processing of a job can be stopped at any moment, another job can be started, and the processing of the preempted job can be restarted where it left of. In other words, the time that was spent processing the preempted job is not lost. Finally, the objective that we consider is the sum of the completion time of all jobs (in the subset).

We will show a universal solution for this problem such that the solution for any subset is at most a multiplicative factor of 2 worse than the optimal schedule

on that subset. This, of course, is also an *a priori* solution of factor 2.

2.1.1 Problem Definition

INPUT: n jobs (with processing times p_j and release dates r_j for $j = 1, 2, \ldots, n$)

OUTPUT: a permutation π of the jobs — the permutation defines a schedule as follows: at every point in time, the first job available of the order is processed

GOAL: minimize $\max \left\{ \dfrac{\sum_{j \in S} \bar{C}_j^{(S)}(\pi)}{\sum_{j \in S} C_j^{(S)*}} \,\middle|\, S \subseteq \{1, 2, \ldots, n\} \right\}$, where $\{\bar{C}_j^{(S)}(\pi)\}_j$ are the completion times of the schedule defined by the permutation π, where the job set is restricted to S, and $\{C_j^{(S)*}\}_j$ are the completion times of the *optimal* schedule for the instance restricted to S.

2.2 Result and Analysis

Claim 8. *Let π be the permutation ordering the jobs in increasing processing time, breaking ties arbitrarily. Then*

$$\max \left\{ \frac{\sum_{j \in S} \bar{C}_j^{(S)}(\pi)}{\sum_{j \in S} C_j^{(S)*}} \,\middle|\, S \subseteq \{1, 2, \ldots, n\} \right\} \leq 2.$$

Proof. Note that it suffices to prove that this ordering is a 2-approximation for $1|pmtn, r_j| \sum C_j$.

Order the jobs such that $p_1 \leq p_2 \leq \ldots \leq p_n$. The following inequality is straightforward:

$$\bar{C}_j \leq r_j + \sum_{k=1}^{n} W_{\text{ALG}}(r_j, \bar{C}_j, k)$$

where $W_{\text{ALG}}(t_1, t_2, k)$ is the amount of time the algorithm spends on processing job k between t_1 and t_2. Between the release date of job j and its completion, no job

$k > j$ will be processed due to the fact that preemption is allowed. Hence,

$$\bar{C}_j \leq r_j + \sum_{k=1}^{j} W_{\text{ALG}}(r_j, \bar{C}_j, k).$$

Of course, one cannot spend more time on a job than its processing time, therefore

$$\bar{C}_j \leq r_j + \sum_{k=1}^{j} p_k.$$

Summing over all j gives and changing the order of summation gives

$$\begin{aligned}
\sum_{j=1}^{n} \bar{C}_j &\leq \sum_{j=1}^{n} r_j + \sum_{j=1}^{n} \sum_{k=1}^{j} p_k \\
&= \sum_{j=1}^{n} r_j + \sum_{k=1}^{n} \sum_{j=k}^{n} p_k \\
&\leq \sum_{j=1}^{n} r_j + \sum_{k=1}^{n} (n-k+1) p_k.
\end{aligned}$$

Now note that we can view the last term as the optimal solution to the modified problem where all job are immediately available. Call the optimal completion times for this problem $C^{(0)}$. The optimal solution to the unmodified problem with release dates is a feasible solution to the modified problem without release dates, therefore

$$\begin{aligned}
\sum_{j=1}^{n} \bar{C}_j &\leq \sum_{j=1}^{n} r_j + \sum_{k=1}^{n} (n-k+1) p_k \\
&= \sum_{j=1}^{n} r_j + \sum_{k=1}^{n} C_k^{(0)} \\
&\leq \sum_{j=1}^{n} r_j + \sum_{k=1}^{n} C_k^{*} \\
&\leq \sum_{j=1}^{n} C_j^{*} + \sum_{k=1}^{n} C_k^{*} \\
&= 2 \sum_{j=1}^{n} C_j^{*},
\end{aligned}$$

where the third inequality follows from the trivial observation that the completion time of a job is after its release date. □

Note that the results straightforwardly carry over for the case of minimizing the *weighted* sum of completion times, when the jobs are ordered by p_j/w_j. Also, we only used the fact that jobs can be interrupted in our analysis, so the result also holds when all time spent on a job is lost when it is interrupted.

Finally, we would like to point out a connection between this chapter and *priority algorithms* as defined by Borodin, Nielsen and Rackoff [9]. Note that if there exists a priority algorithm such that subsets of jobs inherit the priority structure (the priority structure of any subset is induced by priority structure on the full set), then these priority algorithms are universal algorithms with guarantee equal to the approximation guarantee of the priority algorithm.

Chapter 3

AdWords

3.1 Introduction

The emergence of the internet has led to many new and interesting questions within the field of optimization. Moreover, since computer scientists speak the same language, so to speak, (if they are not in the same field already), these problems quickly found their way into the Computer Science Theory and Operations Research Optimization communities.

We will look at one abstraction of a specific problem, faced by Google and other online advertising agencies. The problem we study is the following: queries arrive, and Google (say) needs to determine which advertisement is displayed on the screen (alongside the search results, say). The money that Google receives is equal to the bid that the chosen advertiser made for the query in question. Furthermore, the advertisers have a budget, and cannot be assigned more queries than they can pay for.

We will consider this problem in the *a priori* framework, i.e. all assignment decisions are to be made before any query arrives. This makes sense as the decision about which advertisement to show has to be instanteneous. We will compare our solution to the best possible *a priori* solution, allowing for randomized algorithms, but not for adaptive algorithms.

The motivating paper for looking at this problem is a paper by Mehta, Saberi, Vazirani and Vazirani [25], in which the problem is considered in a purely online adversial (worst-case) setting. Even though that paper's results are mathematically beautiful, they seem to lack in practicality: it seems quite unnatural not to use the

statistical information that is easily obtainable in a digital world. Our main goal is to incorporate this information to get an algorithm that will perform better in practice (compromising the robustness against adverseries).

Note that the problem as described is a simplification of the real world problem, as many of the specifics are simplifications: in practice not one, but multiple advertisements are chosen, the amount paid is assumed to be a first-price auction in our setting, whereas from a game-theoretic perspective this is not preferable, since this type of auction allows participants to "game" the auction, et cetera.

We will assume that queries of each type arrive according to independent Poisson processes. The two main problems that we encounter in our setup, is how we incorporate uncertainty about the estimate of the rates for each query type, as well as how we make sure that the budget for each advertiser is not exceeded. The first problem we will first address in a relaxation of the AdWords problem, which we call Stochastic Matching. For this problem we do not have a budget contraint. Next, we will tackle the second problem, and make sure that no advertiser's budget is exceeded with high probability.

3.2 Stochastic Matching

First we will ignore the budget constraint, and look at Stochastic Matching, focusing on how to deal with uncertainty in regard to the input.

3.2.1 Problem Definition

INPUT: a bipartite graph $G = (V, E)$ with revenues associated with the edges $c : E \to R^{\geq 0}$; not all the vertices will materialize

OUTPUT: a matching (i.e., a subset of E, such that no two edges in the subset are incident to the same node)

GOAL: maximize expected revenue, where revenue for an edge will only be available if both endpoints materialize

3.2.2 Solution With Full Knowledge

Suppose edge e materializes with probability p_e. Then solving

$$\begin{aligned} \max \quad & \sum_e p_e c_e x_e \\ \text{s.t.} \quad & \sum_{e:v\in e} x_e \leq 1 \quad \text{all } v \in V \\ & x_e \in \{0,1\} \end{aligned}$$

gives the optimal solution. (Note that this is an assignment problem (edges to vertices), so we can relax the integrality constraint to $x_e \geq 0$ for all e and still get integral solutions (integrality property).) In other words, we have a poly-time algorithm to solve Stochastic Matching to optimality, given full knowledge of the probabilities $\{p_e\}_e$.

3.2.3 Solution With Uncertainty

Suppose now that we only have an estimate for the probabilities, say $\{\tilde{p}_e\}_e$, such that $\tilde{p}_e \in [(1-\varepsilon)p_e, (1+\varepsilon)p_e]$. How does the optimal solution to the problem with estimated probabilities compare to the optimal solution to the real problem (in the sense of a worst-case comparison)? It turns out that we can get the following result.

Claim 9. *Suppose we are given* $\{\tilde{p}_e\}_e$, *such that* $\tilde{p}_e \in [(1-\varepsilon)p_e, (1+\varepsilon)p_e]$.

Then solving the linear program above, with $\{\tilde{p}_e\}_e$ *replacing* $\{p_e\}_e$ *is a* $(1-2\varepsilon)$*-approximation to the Stochastic Matching instance with full knowledge of* $\{p_e\}_e$*.*

Proof. First of all, note that solving the LP is poly-time, so we have a bona fide algorithm. Now we will show that the approximation guarantee is indeed $(1-2\varepsilon)$.

Let $x^*(u)$ be an optimal solution to

$$\begin{aligned} \max \quad & \sum_e c_e p_e (1+u_e) x_e \\ \text{s.t.} \quad & \sum_{e:v\in e} x_e \le 1 \quad \text{all } v \in V \\ & x_e \in \{0,1\}. \end{aligned}$$

We want to compare $\min_u \left\{ \sum_e c_e p_e x_e^*(u) : u \in [-\varepsilon,\varepsilon]^{|E|} \right\}$ with $\sum_e c_e p_e x_e^*(0)$, the real optimum.

Note that $x^*(0)$ is a feasible solution for any u, so we get

$$\sum_e c_e p_e (1+u_e) x_e^*(u) \ge \sum_e c_e p_e (1+u_e) x_e^*(0) \ge (1-\varepsilon) \sum_e c_e p_e x_e^*(0).$$

Also,

$$\sum_e c_e p_e x_e^*(u) \ge \sum_e c_e p_e \frac{(1+u_e)}{(1+\varepsilon)} x_e^*(u).$$

Therefore,

$$\sum_e c_e p_e x_e^*(u) \ge \frac{1-\varepsilon}{1+\varepsilon} \sum_e c_e p_e x_e^*(0) \ge (1-2\varepsilon) \sum_e c_e p_e x_e^*(0). \qquad \square$$

3.3 AdWords

We will now turn our attention to the AdWords problem.

3.3.1 Problem Definition

Given bids from each bidder i for each query type t, c_{it}, and a budget B_i for each bidder i for some time interval (the same interval for each bidder, without loss

of generality, say of length 1). We want to assign incoming queries to bidders to maximize the (expected) revenue, such that the amount that each bidder has to pay is less than his budget for the time interval.

3.3.2 Assumptions

As stated earlier, we will assume that queries arrive according to independent Poisson processes with parameters $\{\lambda_t\}_t$.

First we will assume full knowledge about these parameters, next we will relax this again.

3.3.3 Algorithm

Solve

$$\begin{aligned} \max \quad & \sum_{i,t} \lambda_t c_{it} x_{it} \\ \text{s.t.} \quad & \sum_i x_{it} \leq 1 \quad \text{all } t \\ & \sum_t \lambda_t c_{it} x_{it} \leq (1-\varepsilon_1) B_i \quad \text{all } i \\ & x_{it} \geq 0 \end{aligned}$$

(ε_1 to be determined later).

Now assign next query of type t to bidder i, such that $x_{it} - f_{it}$ is largest and positive, where f_{it} is the fraction of queries of type t that are assigned to i so far.

We will refer to the LP above as LP(ε_1) for varying values of ε_1.

3.3.4 Analysis

Claim 10. *For an appropriately long "billing" interval $[0, P]$ (P to be determined later), the algorithm gives a feasible solution to the Adwords Problem, that is better*

than $1-\varepsilon$ times the expected value (average case) of the offline optimal solution, with probability $1-\delta$.

Proof. Feasibility: First we use Chebyshev's inequality to bound the probability that there are more than $(1+\varepsilon_2)\lambda_t P$ queries of type t in time P (ε_2 to be determined later). Denote the number of queries of type t that arrive in the interval $[0,P]$ by the random variable $Q_t[0,P]$.

$$P(Q_t[0,P] > (1+\varepsilon_2)\lambda_t P) = P(Q_t[0,P] - \lambda_t P > \varepsilon_2 \lambda_t P) \leq \frac{\lambda_t P}{(\varepsilon_2 \lambda_t P)^2} = \frac{1}{\varepsilon_2^2 \lambda_t P}.$$

So we can choose P large enough such that $Q_t[0,P] \leq (1+\varepsilon_2)\lambda_t P$ for all t with probability $1-\delta$, namely $P \geq 1/(\varepsilon_2^2 \lambda_t \delta)$.

By the way that queries are assigned, we know that the number of queries of type t assigned to bidder i is bounded by $x_{it} Q_t[0,P] + 1$. Therefore each bidder i has to pay at most

$$\begin{aligned}
\sum_t (x_{it}(1+\varepsilon_2)\lambda_t P + 1)c_{it} &= (1+\varepsilon_2)\sum_t \lambda_t P c_{it} x_{it} + \sum_t c_{it} \\
&\leq (1+\varepsilon_2)(1-\varepsilon_1) P B_i + \sum_t c_{it} \\
&\leq (1-\varepsilon_1+\varepsilon_2) P B_i + \sum_t c_{it}.
\end{aligned}$$

Again, we have the freedom to choose P also large enough such that $\sum_t c_{it}$ is "sufficiently small" compared to PB_i (more specifically we will choose P such that $\sum_t c_{it} \leq \varepsilon_3 P B_i$ for ε_3 to be determined later). We conclude that our algorithm in that case gives a solution such that no advertiser is billed more than her budget.

Optimality: We will consider the following LP (call this LP1), and use it to compare the optimal average case *a posteriori* solution and our proposed solution. Let $p_{tj} = P(Q_t[0,P] \geq j)$. The maximum expected revenue, where the budget constraints are interpreted in the "expected value" sense, is the solution to the

following (infinite sized) LP (call this LP1):

$$\begin{aligned} \max \quad & \sum_{i,t,j} p_{tj} c_{it} x_{itj} \\ \text{s.t.} \quad & \sum_{i} x_{itj} \leq 1 \quad \text{all } t, j \\ & \sum_{t,j} p_{tj} c_{it} x_{itj} \leq PB_i \quad \text{all } i \\ & x_{itj} \geq 0 \end{aligned}$$

where for each occurrence of the each query type, we have a separate decision variable.

Subclaim 10.1. *LP1 above has the same optimal value as the LP where the decision variables are restricted to take on the same value for each occurrence of a query type (LP2), i.e.*

$$\begin{aligned} \max \quad & \sum_{i,t,j} p_{tj} c_{it} \bar{x}_{it} \\ \textit{s.t.} \quad & \sum_{i} \bar{x}_{it} \leq 1 \quad \textit{all } t \\ & \sum_{t,j} p_{tj} c_{it} \bar{x}_{it} \leq PB_i \quad \textit{all } i \\ & \bar{x}_{it} \geq 0. \end{aligned}$$

Proof. (LP1) is a relaxation of (LP2), so the optimal value of (LP1) is at least the optimal value of (LP2).

Let $\{x^*_{itj}\}_{itj}$ be an optimal solution to (LP1). Define $\bar{x}^*_{it} = \sum_j p_{tj} x^*_{itj}/(\lambda_t P)$. Now

$$\begin{aligned} \sum_i \bar{x}^*_{it} &= \sum_i \sum_j p_{tj} x^*_{itj}/(\lambda_t P) \\ &= (1/\lambda_t P) \sum_j p_{tj} \sum_i x^*_{itj} \\ &\leq (1/(\lambda_t P)) \sum_j p_{tj} = 1 \end{aligned}$$

for all t, where the change in order of summation is justified by the fact that all terms are nonnegative, and the fact that $\sum_j p_{tj} = \sum_j P(Q_t[0,P] \geq j) = EQ_t[0,P] = \lambda_t P$ is used. Also,

$$\begin{aligned}
\sum_{t,j} p_{tj} c_{it} \bar{x}^*_{it} &= \sum_{t,j} p_{tj} c_{it} \sum_{\ell} p_{t\ell} x^*_{it\ell} / (\lambda_t P) \\
&= \sum_{t,j,\ell} (1/(\lambda_t P)) p_{tj} p_{t\ell} c_{it} x^*_{it\ell} \\
&= \sum_{t,\ell} (1/(\lambda_t P)) p_{t\ell} c_{it} x^*_{it\ell} \sum_{j} p_{tj} \\
&= \sum_{t,\ell} p_{t\ell} c_{it} x^*_{it\ell}
\end{aligned}$$

for all i, so the second constraint holds as well, and the objective function value is the same as the optimal objective function value of (LP1). □

Subclaim 10.2. *(LP2) and the LP used in the algorithm where $\varepsilon_1 = 0$ are equivalent.*

Proof. Again we use the fact that $\sum_j p_{tj} = \lambda_t P$, so the second set of constraints gives us

$$\sum_{t,j} p_{tj} c_{it} x_{it} = \sum_{t} \lambda_t P c_{it} x_{it} \leq P B_i$$

or

$$\sum_{t} \lambda_t c_{it} x_{it} \leq B_i$$

for all i.

Similarly, the objective is the same up to a multiplicative factor. □

Subclaim 10.3. *An optimal solution to the LP used in the algorithm is an $(1-\varepsilon_1)$ approximate solution to the optimal solution of (LP1).*

Proof. Let $\{x^*_{ij}(0)\}_{ij}$ be an optimal solution to the LP(0). Then $\tilde{x}^*_{ij}(\varepsilon) := x^*_{ij}(0)(1-\varepsilon_1)$ is a feasible solution to LP with parameter ε_1, with objective value $(1-\varepsilon_1)$

times the optimal objective value of LP(0). So the optimal value of LP(ε_1) is at least $(1-\varepsilon_1)$ times the optimal objective value of LP(0). Since this solution is also feasible for LP(0), we have that an optimal solution to LP(ε_1) is an $(1-\varepsilon_1)$ approximate solution to the optimal solution of LP(0), and hence to (LP1). □

We still have to show how the optimal solution to (LP1) and the optimal average case *a posteriori* solution compare. Fix the length of the interval, say P. Let I be a realization of the query arrivals, say $q_t(I)$ is the number of queries of type t for I. The *a posteriori* optimum can be found by solving the following integer program:

$$\begin{aligned} \max \quad & \sum_{i,t,j} c_{it} x_{itj} \\ \text{s.t.} \quad & \sum_{i} x_{itj} \leq 1 \quad \text{all } t,j \\ & \sum_{t,j} c_{it} x_{itj} \leq PB_i \quad \text{all } i \\ & x_{itj} \in \{0,1\}. \end{aligned}$$

Let $\{x^*_{itj}(I)\}_{itj}$ be optimal solutions for all possible I. Define

$$\tilde{x}^*_{itj} = \sum_I P(I \text{ occurs}) x^*_{itj}(I)$$

for all i,t,j.

Subclaim 10.4. *$\{\tilde{x}^*_{itj}\}_{itj}$ is a feasible solution to (LP1) above.*

Proof. The first set of inequalities trivially hold for $\{\tilde{x}^*_{itj}\}_{itj}$. For the second set of inequalities note that $p_{tj} = \sum_{I:q_t(I)\geq j} P(I \text{ occurs})$ and switch the order of summation. □

Noting again that a linear program can be solved in polynomial time in the size of the input, we have established that the algorithm is at most a factor $(1-\varepsilon_1)$ worse than the expected value (average case) of the offiline optimal solution. □

Corollary 11. *For an appropriately long "billing" interval* $[0, P]$*, the algorithm gives a feasible solution to the AdWords Problem, that is within* $1-\varepsilon$ *of the best a priori algorithm, with probability* $1-\delta$*.*

Proof. Since the best *a posteriori* algorithm is better than any *a priori* algorithm, we also have that the algorithm is at most a factor $(1-\varepsilon)$ worse than any *a priori* (randomized) algorithm. □

Finally, a note on the parameters P, ε, δ, ε_1, ε_2 and ε_3.

Relationship between parameters: Obviously we want $\varepsilon_1 := \varepsilon$. Let n be the number of query types. From the union bound over all query types, we get that we want

$$\frac{n}{\varepsilon_2^2 \lambda_{min} P} \leq \delta,$$

where $\lambda_{min} = \min_t \lambda_t$ (yes, one can make this slightly stronger).

We also want $\sum_t c_{it}/PB_i \leq \varepsilon_3$ for all i, and $1-\varepsilon+\varepsilon_2+\varepsilon_3 \leq 1$, i.e. $\varepsilon_2+\varepsilon_3 \leq \varepsilon$.

Choosing, rather arbitrarily, $\varepsilon_2 = \varepsilon_3 = \varepsilon/2$, we get

$$P \geq \max\left\{\frac{4n}{\varepsilon^2 \lambda_{min} \delta}, \max_i \left\{2 \sum_t c_{it}/(B_i \varepsilon)\right\}\right\}.$$

Another way that the relationship between the parameters can be used, is figuring out what ε can be achieved, given the length of the billing period.

3.3.5 Uncertainty with respect to $\{\lambda_t\}_t$

We can treat this uncertainty in a similar way as we treated it for the Stochastic Matching problem. We just have to scale the budget by $1/(1+\varepsilon)$, and therefore we will lose another ε.

Chapter 4

Creating Nearly Isogenic Lines

4.1 Introduction

The discovery of deoxyribonucleic acid (DNA) opened up a whole field of research in science (and biology particular): what do the different parts of the genetic material do? Or, less broadly, can we point out parts of the genome as the cause of (or at least having an influence on) certain traits (by which I mean quantifiable features of an organism)?

Many researchers are working on problems related to this question. We look at a problem that is the natural followup to identifying a part of the DNA that is of interest: How can we isolate this little piece of genetic material, and insert it into a DNA background of choice?

More precisely, assume for a moment that we have unlimited access to two different types of the same organism, one type we will call "wild", the other "domesticated". We are interested in isolating little pieces of the wild DNA, i.e., we will try to create an individual organism that has domesticated DNA everywhere, except for a small contigious piece. These are what are called (heterozygous) **Nearly Isogenic Lines** or NILs.

This problem is the topic of this chapter. We will now first start with a short introduction to the relevant biology, and define the terms some of which we could not avoid using already. In the next few sections we will create a mathematical framework for the problem, after which we will propose an approach to creating NILs, while trying to minimize the number of individuals that have to be grown or bred. The approach is applied to a real data set. Finally, as this approach does not

pretend to be the definitive answer to the problem, the chapter concludes with a discussion on the caveats of the proposed approach, and directions for improvement and further research.

4.2 Very Short Introduction to the Relevant Biology

We are interested in what happens when we cross (mate) individuals. Let us first look at what biologists know about this subject.

Genetic information in all known cellular life is stored in big molecules called deoxyribonucleic acid (DNA). It is divided up into several chromosomes (these are nothing but large continuous pieces of DNA). Note that these chromosomes are not "interchangeable" in any way: each of them contains some specific part of the genome. Also note that the number of chromosomes depends on the species.

Each (normal) individual has 1 or more sets of chromosomes, the number most commonly one (*haploid*) or two (*diploid*), but there exist species which have a larger number. Cells of sexually reproducing species have 2 sets of chromosomes, one set coming from the mother and one set coming from the father. We will from here on restrict our attention to diploid species.

A child is created from two cells, one from the mother and one from the father, that are specifically created for this purpose. Each of these cells, called *gametes*, contain only one set of chromosomes (i.e., they are haploid). They are produced through a process called *meiosis*, in which one set of chromosomes is created from the two of the parent.

Let us assume now that the species that we are interested in has $2n$ chromosomes (or n chromosome pairs, as we are assuming that the species is diploid). Further, let us assume that all chromosomes of both mother and father are dis-

tinct. Then, just by the fact that for each chromosome pair, a child inherits one chromosome from its mother, and one from its father, there are 2^{2n} distinct genetic possibilities for the child (for each chromosome there are 2 possible sources).

But it is even more interesting: during meiosis, the chromosomes that are formed are not necessarily either of the chromosomes of the parent. There may occur *crossovers*: this means that the chromosome that is produced is actually a patchwork of the two chromosomes of the particular parent. (The idea here is that contiguous pieces of DNA of one chromosome are replaced with contiguous pieces of DNA of the other chromosome, that is (roughly) in the same position, so that the resulting chromosome should be a "working" chromosome again.) I will now make this more precise, and introduce the assumptions that I will be using throughout this chapter.

As is generally known now, one can succintly represent a chromosome by a long string, where each character is either $\begin{matrix}A\\T\end{matrix}$, $\begin{matrix}T\\A\end{matrix}$, $\begin{matrix}C\\G\end{matrix}$ or $\begin{matrix}G\\C\end{matrix}$. These letters represent the two *nuleotides* at the position of the DNA: adenine, cytosine, guanine and thymine. A good way to envision the crossover process is as follows: Imagine two of these strings (of equal length) lying on top of each other. One of the strings is selected at random, and the new chromosome will consist of the DNA of this string, until a crossover occurs. At that point the DNA of the other string will be copied, until the next crossover, *et cetera*.

To fully specify the process, we must now answer the following questions: what is the probability that a particular chromosome is chosen at the start and what is the probability distribution on the crossover positions?

These are, of course, both biological questions, and the true answers are —to the best of my knowledge— not known (the idea that both chromosomes have

equal probability of being chosen at the start seems a natural one, though).

Because we will use a simulation based approach, we would be content with a black box that provides us samples of these processes. For ease of exposition the focus will be on one particular model, due to Haldane [19]. It is an easy model and the most commonly used model. It will be made clear when this model is used explicitly in the next sections. Also we will note how this can be changed using simulations and black box access to obtain samples of the crossover process.

Haldane's idea is that there exists a transformation of the physical layout of a chromosome to the real line, such that the crossover points form a Poisson process with parameter 1. For each chromosome (and each species) this transformation is (potentially) different.

Note that there are biological objections to this model, first and foremost the fact that it is possible in this model to have crossovers that are very close together. There is a biological reality named *interference* that says that crossovers that are close actually do not occur. We will ignore this fact.

Also, in real life *mutations* occur with very small probability, these are spontaneous changes in the DNA. We will ignore this too.

From here on, we will not worry about the (messy) physical reality, and will think of the genetic material as being on a line, where recombinations (crossovers) occur according to a Poisson process with parameter 1.

4.3 Backcrossing

Now that we know a little more about the biology of crossing, and have settled on the model that we are going to use, we can specify our assumptions and setup more precisely.

We are assuming for the moment that we are starting with an unlimited supply of two different types of the same species, i.e., we are assuming that for each type we have an unlimited supply of individuals that are *genetically identical* (we will revisit this assumption very shortly). Further, these types are assumed to be such that when crossing two of the same type, another genetically identical individual will be the result. Such a type is called an *inbred line* in genetics.

Note that this implies that the two chromosomes of each pair of chromosomes for an inbred individual are identical as well.

When crossing two individuals, one of each of these types, the resulting children are fully determined as well: for each pair of chromosomes, they will have one chromosome that is identical to the corresponding chromosome of each of the parents. We will call these individuals the *F1 population*.

The situation becomes more interesting when we start crossing individuals from the F1 population. There are multiple possibilities now: individuals from the F1 population can be crossed with other individuals from the F1 population, giving rise to what is called an *F2 population*.

Alternatively, individuals from the F1 population can be crossed with an individual that is genetically identical to one of its parents. This is what is called *backcrossing*. The individuals resulting from this are called *BC*, or even *BC1*. The "1" in BC1 is suggestive, and, indeed, the BC1 individuals can be backcrossed again, and create a *BC2*-population, *et cetera*.

Note that for backcrosses, one of the chromosomes of each pair is identical to that of the parent which is used in the backcrossing. The other chromosome is a "patchwork" of the chromosomes of both parents.

We assume that the only crosses that we are allowed to make are backcrosses.

We will call the type of the parent which is used over and over again in the crosses, the "domesticated" type, and the type of the other inbred line the "wild" type.

The rationale for only considering backcrosses is threefold. First of all, these crosses are relatively easy to analyze: for each location on the genome there are only two possibilities, an individual has either DNA that came from the domesticated line for both chromosomes (is *homozygous*), or it has DNA from the domesticated line on one chromosome, and from the wild line on the other (is *heterozygous*). For a trait that is scored as a number, this means that there can only be 3 possibilities: there is no difference given the DNA, the *phenotype* (the observed trait value) of the homozygous individuals is higher or the phenotype of the heterozygous individuals is higher. When allowing for all possible DNA combinations, there are already 6 possibilities ignoring the fact that phenotypes can be the same.

Secondly, we really only need to have an unlimited supply of the "domesticated" variety, relaxing the earlier assumption. (Here we do assume that the "wild" individual can create sufficient F1-offspring.)

Finally, our goal of getting individuals with only one small contiguous part of wild DNA, suggests that backcrossing with the domesticated type would be the right approach.

4.4 Terminology

Definition 12. *Points on the genome for which it can be deduced whether it came from the domesticated or wild DNA are called* markers. *We assume that we have knowledge of the* position *of the marker on the genome (in terms on our probabilistic map).*

Definition 13. *An* interval *is the DNA between two consecutive markers on one*

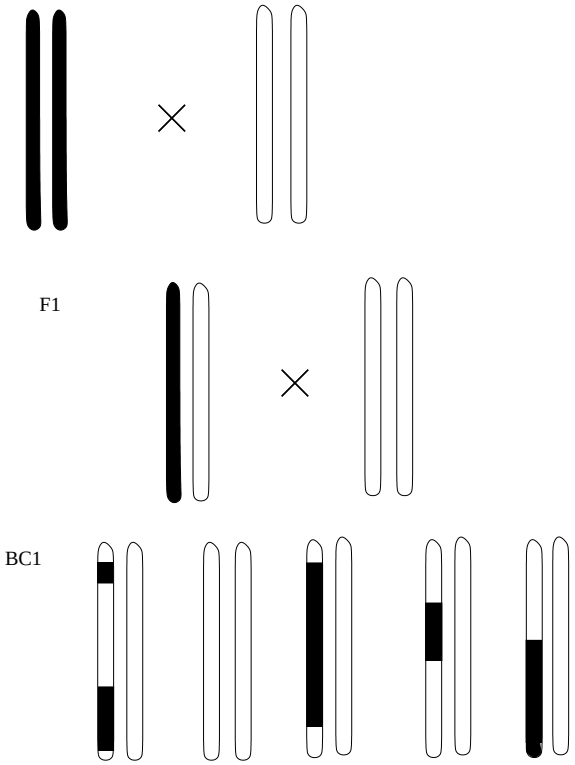


Figure 4.1: Cartoon representation of the inheritance of genetic material. Wild type DNA is depicted as black, domesticated type as white. First a fully homozygous wild individual is crossed to a fully homozygous domesticated individual, the result being a fully heterozygous individual. This individual is *back crossed* to the fully domesticated parent, giving rise to (potentially) all kind of genetically different children.

chromosome.

Definition 14. *Given a set of markers, say $\mathcal{M}$, we can now define* individuals *as functions from $\mathcal{M}$ to $\{0, 1, ?\}$, where a 1 indicates wild DNA for a particular marker, a 0 indicates domesticated, and a ? indicates that it is unknown whether the marker has wild or domesticated DNA.*

Note that this setup does not allow a conclusion about every particular interval, since a crossovers can take place anywhere in an interval.

The following definition defines what we are trying to create.

Definition 15. *Given an upperbound u, define a plant to be a u-*success *for interval i, if the plant has domesticated DNA everywhere, except for a contiguous stretch of length at most u, that includes the two markers defining interval i. (If u is smaller than the length of interval i, then a plant is a u-*success *if the plant has domesticated DNA everywhere except for a contiguous stretch that includes just the two markers defining interval i, and no other markers.)*

We caution the reader that we will usually suppress the u, even though successes are always defined relative to some u. The context should make clear what the value of u is.

4.5 Problem Description

The problem under consideration is the following.

INPUT:

- a set of *markers*, say $\mathcal{M}$,

- a set of *individuals*, $\mathcal{I}$, where the DNA is (mostly) known for the set of markers,
- a bound on the number of generations allowed, m,
- an accepted probability of failure, α,
- an upperbound u on the maximum allowable length of an interval

OUTPUT:

- a *growing strategy* such that among the individuals grown, there exist u-*successes* for all *intervals* with probability at least $1 - \alpha$, and such that at most m generations of individuals are grown

OPTIMIZATION CRITERION:

- minimize the total number of individuals grown.

A *growing strategy* is any prescription (algorithm) that decides which crosses are to be made.

Note that *individual* and *plant* are used interchangeably, even though nothing in this section is particular for plants.

4.6 Modus operandi

4.6.1 Unique lines of descent only

We will restrict ourself to *nonadaptive* strategies, i.e., the whole growing scheme will be decided on a priori, and will not be influenced by (knowledge about) the intermediate plants we grow. This also means that we do not have to sequence the DNA of intermediate plants.

In our first attempt to solve this problem, we will restrict our strategies even further, disallowing any two plants to share the same (not fully domesticated) parent (except for the children from the base set of plants). This has only mathematical reasons, making the problem easier to solve: we can now succintly summarize the strategy as $\{x_{gj}\}_{gj}$, where x_{gj} indicates the number of plants we will grow for g generations, starting with base plant j (and a fully domesticated plant).

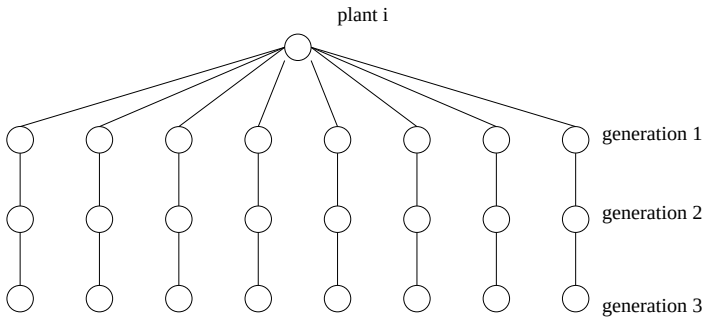


Figure 4.2: Unique lines of descent

We will relax this last restriction in the next subsection. Further research is required to see whether the other assumption (nonadaptivity) can be relaxed, and/or how many extra plants are needed because of these restrictions.

Under these restrictions it is easy to estimate the probability of a success for every plant in the base set, every number of (allowable) generations and every interval, using *(Monte Carlo) simulation*. Let us call the probability of a success for interval i, for a particular g-th generation descendent of base plant j p_{igj}. The probability of having a success for interval i given a growing strategy can now be expressed as

$$1 - \prod_{gj} (1 - p_{igj})^{x_{gj}}.$$

Note that for a fixed generation g and fixed base plant j the probabilities p_{igj} actually constitute something akin to a multinomial distribution. When assuming

that u is smaller than twice the length of the shortest interval, and a final poba-bility $q_{gj} = 1 - \sum_i p_{igj}$ is added, indicating the probability of no success for any interval, this is exactly a multinomial distribution. In order to make the problem more tractable, however, we will ignore the dependence of this "multinomial-like" distribution and pretend we are dealing with independent Bernoulli distributions for each interval.

Claim 16. *Assume that u is smaller than twice the length of the shortest interval. If the nonzero x_{gj} are of reasonable size and the probabilities p_{gj} are small, then ignoring the fact that we are actually dealing with a multinomial distribution, is a reasonable thing to do.*

Proof. We will focus on two intervals, say interval 1 and 2. Fix a growing strategy $\{x_{gj}\}_{gj}$. Denote by C^i the event that a success for interval i occurs, and by $S(\{k_{gj}\})$ the event that exactly k_{gj} successes for interval 2 occur, starting with base plant j in generation g, for each g and j.

$$\begin{aligned}
\mathbb{P}(C^1 \cap C^2) &= \sum_{\{k_{gj}\}_{gj}: 0 \le k_{gj} \le x_{gj}, \sum_{gj} k_{gj} \ge 1} \mathbb{P}(C^1 | S(\{k_{gj}\})) \mathbb{P}(S(\{k_{gj}\})) \\
&= \sum_{\{k_{gj}\}_{gj}: 0 \le k_{gj} \le x_{gj}, \sum_{gj} k_{gj} \ge 1} \big(1 - \prod_{g,j} \big(1 - p_{1gj}\big)^{x_{gj} - k_{gj}}\big) \prod_{g',j'} p_{2g'j'}^{k_{g'j'}} \\
&\le \sum_{\{k_{gj}\}_{gj}: 0 \le k_{gj} \le x_{gj}, \sum_{gj} k_{gj} \ge 1} \big(1 - \prod_{g,j} (1 - p_{1gj})^{x_{gj}}\big) \prod_{g',j'} p_{2g'j'}^{k_{g'j'}} \\
&= \big(1 - \prod_{g,j} (1 - p_{1gj})^{x_{gj}}\big) \sum_{\{k_{gj}\}_{gj}: 0 \le k_{gj} \le x_{gj}, \sum_{gj} k_{gj} \ge 1} \prod_{g',j'} p_{2g'j'}^{k_{g'j'}} \\
&= \big(1 - \prod_{g,j} (1 - p_{1gj})^{x_{gj}}\big) \mathbb{P}(C^2) \\
&= \mathbb{P}(C^1)\mathbb{P}(C^2).
\end{aligned}$$

So assuming indepence does not underestimate the probability of joint successes. We will now try to bound the error. We can derive a lower bound using a similar

line of reasoning. Assume without loss of generality that $x_{gj} > 0$ for all g and j.

$$\begin{aligned}
\mathbb{P}(C^1 \cap C^2) &= \sum_{\{k_{gj}\}_{gj}: 0 \le k_{gj} \le x_{gj}, \sum_{gj} k_{gj} \ge 1} \big(1 - \prod_{g,j} \big(1 - p_{1gj}\big)^{x_{gj} - k_{gj}}\big) \prod_{g',j'} p_{2g'j'}^{k_{g'j'}} \\
&\ge \sum_{\{k_{gj}\}_{gj}: 0 \le k_{gj} \le x_{gj}, \sum_{gj} k_{gj} = 1} \big(1 - \prod_{g,j} \big(1 - p_{1gj}\big)^{x_{gj} - k_{gj}}\big) \prod_{g',j'} p_{2g'j'}^{k_{g'j'}} \\
&= \sum_{g'j'} \big(1 - \prod_{g,j} \big(1 - p_{1gj}\big)^{x_{gj} - \chi\{g=g', j=j'\}}\big) p_{2g'j'} \\
&\ge \sum_{g'j'} \big(1 - \prod_{g,j} \big(1 - p_{1gj}\big)^{x_{gj} - 1}\big) p_{2g'j'} \\
&= \big(1 - \prod_{g,j} \big(1 - p_{1gj}\big)^{x_{gj} - 1}\big) \sum_{g'j'} p_{2g'j'} \\
&\ge \big(1 - \prod_{g,j} (1 - p_{1gj})^{x_{gj} - 1}\big) \big(1 - \prod_{g,j} (1 - p_{2gj})^{x_{gj}}\big) \sum_{g'j'} p_{2g'j'} \\
&= \mathbb{P}(C^1)\mathbb{P}(C^2) \frac{1 - \prod_{g,j} (1 - p_{1gj})^{x_{gj} - 1}}{1 - \prod_{g,j} (1 - p_{1gj})^{x_{gj}}} \sum_{g'j'} p_{2g'j'}.
\end{aligned}$$

By assumption, the x_{gj} are of reasonable size, and thus the fraction will be close to 1.

We conclude that pretending that the success events are independent is a reasonable thing to do as long as $\sum_{g'j'} p_{2g'j'} \approx 1$. □

We therefore approximate a success for all intervals as

$$\prod_i \big(1 - \prod_{gj} (1 - p_{igj})^{x_{gj}}\big).$$

Note that there is actually another source of "noise" for this approximation: we do not have our hands on the real probabilities, but only on the results of Monte Carlo simulations. We will ignore this, noting that we it is not computationaly expensive to run lots of similations to get high quality estimates of the probabilities. (One may elect to take a conservative estimation of the probabilities, say the lower end of a confidence interval, to accommodate for this — we did not choose to do this.)

We are thus interested in solving the following program:

$$\begin{aligned} \text{minimize} \quad & \sum_{jg} g x_{gj} \\ \text{such that} \quad & \prod_{i} \left(1 - \prod_{gj} (1 - p_{igj})^{x_{gj}}\right) \geq 1 - \alpha \\ & x_{ig} \in \mathbb{Z}^{\geq 0}. \end{aligned}$$

In order to make this an *integer linear programming problem*, we break up the constraint into multiple constraints, one for each interval:

$$\begin{aligned} \text{minimize} \quad & \sum_{jg} g x_{gj} \\ \text{such that} \quad & 1 - \prod_{gj} (1 - p_{igj})^{x_{gj}} \geq 1 - \beta_i \text{ for each } i \\ & \prod_{i} (1 - \beta_i) \geq 1 - \alpha \\ & x_{ig} \in \mathbb{Z}^{\geq 0}. \end{aligned}$$

Now we can take logarithms for the first family of constraints and finally, to get rid of the second constraint, we quite arbitrarily set $\beta_i = (1 - \alpha)^{1/n}$. We are left with a bona fide integer linear program:

$$\begin{aligned} \text{minimize} \quad & \sum_{jg} g x_{gj} \\ \text{such that} \quad & \sum_{gj} \log(1 - p_{igj}) x_{gj} \leq \frac{1}{n} \log(1 - \alpha) \text{ for each } i \\ & x_{ig} \in \mathbb{Z}^{\geq 0}. \end{aligned}$$

We proceed as follows:

1. Probabilities are estimated using Monte Carlo Simulation.

2. The integer linear program that uses these probabilities is solved using CPLEX (without the integrality constraint).

3. The resulting growing strategy is validated using Monte Carlo Simulation.

Results

We tested this approach on our first data set (see appendix), using $\alpha = -3$ as parameter, which corresponds to an overall confidence level of about 95%.

In short: about 100,000 to 200,000 plants are needed in the growing strategy. We will not go into more detail, as these numbers are not very precise for a lot different reasons.

The data we got has missing marker information data. In the first experiments we ran, we were not very sophisticated in the treatment of the data: We assumed that the DNA changed from being domesticated to being wild (and vice versa) at the particular marker where the change occurred. Also, we treated missing marker data as being domesticated markers.

Later we used crossover points as the data in the simulations: First we simulated the DNA of the base set plant, based on the marker data (note that missing marker data is now easily handled), and next we simulated crossover points for the next generations.

We will omit the details of these procedures as these are readily filled in by anyone who has some experience in Monte Carlo simulation. We do want to note one thing however: simulating the base set plant over and over again skews the probabilities. There again is dependence that we are conveniently ignoring: namely there is exactly one realization of the base set plant. We did not persue this particular direction of research, but focused instead on getting better (less restricted strategies).

4.6.2 k-ary trees of descent

It seems rather wasteful to have only unique lines of descent. One would expect that the number of plants in the growing strategy may be drastically reduced when intermediate plants are allowed to have multiple children, as the examples in figure 4.2 and 4.3 illustrate.

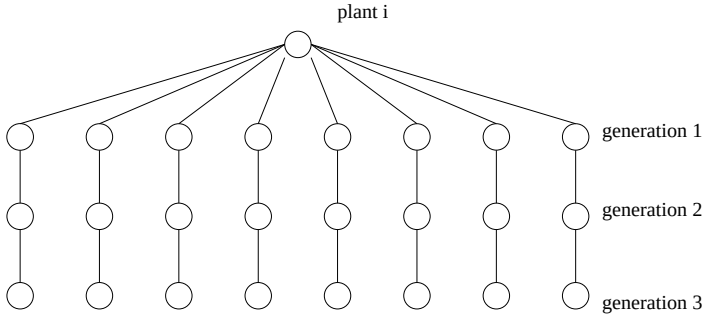


Figure 4.2: Allowing growing strategies with unique lines of descent only, will make the number of plants grow linearly in the number of generations

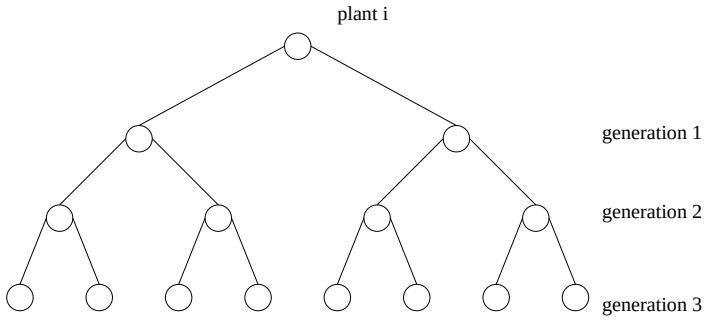


Figure 4.3: Allowing growing strategies that are k-ary trees, have a number of intermediate plants that is linear in the number of plants in the final set, independent of the number of generations

It is a priori unclear, whether the probabilistic dependence between the plants in the final set that we introduce using growing strategy that are k-ary trees, will have big influence on the quality of the solution (i.e., whether the final set produced

has the properties that we want with sufficiently high probability).

We experimented with just turning the growing strategies that we produced in the preceding section into k-ary trees that produce the same number of plants in the final generation. When validating these strategies, we found that the quality deteriorated badly. For instance, an interval that was only covered in 83% of simulation runs, while aiming for total success probability of 99.5%. The reduction in the number of plants is however very significant. This motivates the following.

Fix a plant in the base set and a particular interval for which the base plant has wild DNA, say i. Let Y_g^ℓ be the event that the ℓth plant in generation g has wild DNA for i, and let X_g^ℓ be the event that the ℓth plant in generation g is a success for interval i.

Define $r = \mathbb{P}(Y_g^\ell | Y_{g-1}^{p(\ell)})$, where $p(\ell)$ is the plant in the $(g-1)$st generation that is the parent of the ℓth plant in generation g.

It is not hard to see that r does not depend on ℓ or g: r can be seen as the probability that the particular interval is from a specific parent. The probability of this is equal to the probability that at the start of the interval the DNA is from the specific parent, times the probability that there will not be a crossover. (So, $r = (1/2)\exp(-\text{length interval})$.

Also define $q_g = P(X_g^\ell | Y_g^\ell)$. Note that as $\{Y_g^\ell\}_\ell$ are independent and identically distributed for each g, q_g does not depend on ℓ (it does depend on g).

Now $\mathbb{P}(X_g^\ell) = q_g r^g$. But, more interestingly, we can express the probabilities that we are interested in in r and q's, and we can estimate the values of q using Monte Carlo simulation.

Claim 17. *Suppose that some gth generation plant has wild DNA for interval i. Then the probability that none of the offspring of this plant, d generations down*

(so generation $g+d$), is a success for interval i, is

$$f^d(g) = \big((1-r) + r(f^{d-1}(g+1))\big)^k$$

if $d \geq 2$, where $f^1(g) = \big((1-r) + r(1-q_g)\big)^k$.

Proof. By induction on d. For $d = 1$, the probability that the children of the plant do not have wild DNA for this interval are i.i.d. with probability $(1-r)+r(1-q_g)$, immediately giving the result.

Suppose the claim holds for $d = m$. Let us calculate the probability for $d = m+1$. Let us consider one child of the plant. The probability that this plant has wild DNA for the particular interval is r. By the induction hypothesis we know that the probability that the k-ary tree with this child as the root of depth m has probability $f^m(g+1)$ of not having an offspring m generations down (so generation $g+m+1$), being a success for interval i. So for each child the probability is $(1-r) + r(f^{d-1}(g+1))$ and these events are independent. □

We thus have a way to relax our restriction on growing strategies to growing strategies that are trees. To decrease the number of possibilities somewhat, we will only consider k-ary trees, trees for which every node (except the leaves) have k children.

We have the following integer linear program:

$$\begin{aligned} \min \quad & \sum_{gkj} \big(\sum_{\ell=1}^{g} k^\ell\big) x_{gkj} \\ \text{s.t.} \quad & \sum_{gkj} \log(f^g_{ikj}(0)) x_{gkj} \leq \frac{1}{n} \log \alpha \quad \forall i \\ & x_{gkj} \in \mathbb{Z}^{\geq 0}, \end{aligned}$$

where f^g_{ikj} is the f^d function defined above for interval i, k-ary trees and base set plant j.

Simulating q's

Given all the crossover points in the DNA of a plant, it is easy to do Monte Carlo simulations of the offspring when the particular plant is backcrossed.

We are faced with a slight complication, though: we do not know the exact positions of the crossover points, we only know the DNA at the markers. We therefore need to be able to sample from the all possible crossover point configurations, given the DNA at the markers.

This is also not too hard, once one realizes that the position of the crossover points, conditional on the number of them, say ℓ, can be simulated by ℓ uniformly distributed random variables, see for example Theorem 1.2.5 in [27].

Also note that the number of times that we need to simulate the crossing of each of the plants in the base set is independent of the number of intervals: we can use this information to get the estimates of the q's that correspond to the particular plant for all intervals. Further note that because these simulations are done for each plant independently, this step in the algorithm can be done relatively fast when using parallel computers.

Results

Not every interval is created equal: to get a succes for some of the intervals we need many more plants than for some other intervals. This has different reasons: An interval may not be covered by many different plants in the base set, or the plants that have wild DNA for the particular interval, have a lot of wild DNA, which makes isolating the interval harder.

Also the length of an interval, and the length of neighboring intervals have an impact on the probability of creating a success. More in particular, if neighboring

intervals are short, then the probability of getting a crossover in this neighboring interval is small, and hence the probability of success for the interval is small.

We tried to remedy this problem by relaxing the requirement of having a success for all intervals, and instead requiring having a success for all intervals except U intervals (by adding additional binary decision variables to the linear program, which now became a mixed integer linear program). We also added a constraint on the number of generations. The optimization results for our data set are given in Table 4.1. To obtain the estimates of the probabilities, we ran 100,000 simulations of backcrosses for each plant, for up to 6 generations. We set $\beta = -10$, giving a confidence level that was a little better than 99.5% (to be precise, we had a theoretical confidence level of $(1 - \exp(-10))^{102} \approx 99.54\%$).

There are 14 intervals that are not covered by the any plant in the base set. U reflects the number of additional intervals that are not covered by the solution.

Table 4.1: Number of plants needed to get 30 cM-successes for all but U intervals in at most gen generations.

gen \ U	0	1	2	3	4	5
1	1,050,819	743,802	691,715	670,660	651,087	643,318
2	115,123	99,044	88,273	79,034	73,712	73,068
∞ (3)	79,876	64,748	56,673	53,099	50,904	50,315

Validation

Validation of the results indicated that using the growing strategies obtained from the linear program, gave mostly the desired results. The probability that there existed an interval that was not a success, however, turned out to be around 3%,

significantly higher than the desired .5%. More in particular there were a few intervals for which the probability of a success turned out to be significantly lower than what we were aiming for.

A possible reason for this might be the following. As noted earlier, there is a subtle problem with our approach: In estimating the q's, we are sampling a possible DNA configuration for the base set plant every time that we start a new simulation of the offspring. In reality there is only one realization, and therefore there is more dependence that we are ignoring.

We therefore tried the following enhancemant of the model: we simulated the probability that a particular base set plant indeed has wild DNA for a particular interval, and added the constraints that the probability that at least one of the base set plants that are chosen for get a success for a particular interval is lower bounded by some parameter.

As this entailed adding more binary decision variables, and quite a few at that — namely the number of plants in the base set times the number of intervals — this did not improve the running time of the algorithm. But what was worse, is that the validation indicated that this model did not give better results: the cure seemed actually worse than the disease.

4.7 Conclusion, extensions, future work

We have presented the first optimization approach for one important problem in biology. The results that we obtained seem to be very impressive to the biology community. It must be noted however, that we had to make some simplifying assumptions that need further scrutiny.

The obvious caveat of the results presented in this chapter, is the fact that we

ignore dependence structure that is present in the probability distributions of the successes. It would be great to get rid of one or more of these, but I do not see an easy way around this.

Quantifying the effect of the fact that we are using estimations of probabilities, instead of the true probabilities, is another goal. As is doing some smart with the β_i's instead of setting them all equal to $(1-\alpha)^{1/n}$. (We did try the more or less obvious trick of iterative improvement, pretendeding that the probabilities are variables too. This did not work well, as the method did not converge for the cases that we were interested in.)

An easier modification (and improvement) would be the following. Note that in the linear programs that we use, the plants that we grow for creating a success are disjoint for each interval, whereas "intermediate" plants could (and maybe should) be "shared". It is not hard to see that one can introduce additional variables and get around this problem. Also relaxing the fact that only k-ary trees are allowed should prove helpful.

I also want to note the following about the fact that some intervals needed many plants in comparision with other intervals. Some of these seem to be an artifact of the way we chose to relax our notion of success to U-success: the most costly intervals turned out to be the ones that were relatively large (slightly smaller than U), with very short neighboring intervals on each side (so that the total length of the interval plus one neighbor was slightly more than U). This is hardly surprising, but it would be a good to address this when adopting our approach: The selection of markers can have a great impact on the optimization results.

Finally, the big open question is whether it would be possible to come with an adaptive strategy. The obvious approach is a Stochastic Dynamic Program, with

a prohibitively large computing time (incidently, the topic of the next chapter is shortcutting the running time of Dynamic Programs with a special structure — a structure that the Stochastic Dynamic Program for this problem lacks). Also tradeoffs between the cost of genotyping and the (expected) added benefit of genotyping would then come into play.

This is most certainly not the last word on this topic, as there are many opportunities for further research.

Chapter 5

Approximate Dynamic Programming

5.1 Introduction

Every day derivatives of assets are traded (for instance at stock markets), and decisions have to be made regarding the fair price of those derivatives. We will study this pricing problem, and develop a fully polynomial-time approximation scheme to estimate the fair price arbitrarily closely (under certain assumptions). Our approach is based on transforming a standard stochastic dynamic programming method (which is not polynomial-time) by a new state-space sparsification technique to yield an efficient algorithm. We believe this approach will have many other applications.

More specifically, consider the following problem. Suppose we have a model for the future price of an asset over a discrete finite time horizon: for each time t, we have a probability distribution on the difference between the current price of the asset and its price in the next time step, $\{p_\Delta^{(t)}\}_\Delta$. Furthermore, for each time t, we have a function that gives the payoff of the option if we exercise it at time t, given the stock price s_t at time t, say $G_t(s_t)$. We can exercise the option at most once.

Finally, we assume that the option has some expiration time T, that is, there is some finite time T after which the option has no value: $G_t(\cdot) = 0$ for all $t > T$.

So, at each time t, the decision has to be made whether or not to exercise the option at that particular time. If it is exercised, then the payoff is the payoff that the option generates now, given the stock price s_t, i.e., $G_t(s_t)$. If one chooses not to exercise the option at time t, the expected payoff can be calculated as the expected value of the option at time $t + 1$, where the expectation is taken over the stock

price at time $t+1$. More precisely, let $V_t(s_t)$ be the value (or price) of the option at time t, given that the stock price is s_t at time t, and let S_t be (the random variable representing) the stock price at t; we then get the following recursive definition for the fair price of the option at time t:

$$V_t(s_t) = \max\{G_t(s_t), E_{S_{t+1}}[V_{t+1}(S_{t+1})|S_t = s_t]\}$$

for each $t = 0, 1, \ldots, T$ and $V_{T+1}(\cdot) = 0$ (since $G_t(\cdot) = 0$ for all $t > T$). (We use the notation $E_S[X|Y]$ to denote the expectation of the random variable X, conditioned on Y, with respect to the probability distribution over the random variable S.) Using the assumption on the development of the stock price, we can rewrite this as

$$V_t(s_t) = \max\{G_t(s_t), \sum_{\Delta_{t+1}} V_{t+1}(s_t + \Delta_{t+1}) p^{(t)}_{\Delta_{t+1}}\}$$

for each $t = 0, 1, \ldots, T$ and $V_{T+1}(\cdot) = 0$.

We are interested in calculating $V_0(s_0)$ for a given initial stock price s_0. The obvious way to try and find this value, is to use stochastic dynamic programming — first one calculates and stores $V_T(\cdot)$ for all possible values of s_T, and next one uses these to calculate and store $V_{T-1}(\cdot)$ for all possible value of s_{T-1}, and so forth. The problem with this approach is that there might be an exponential number of possible values for the stock price at time T, i.e., this approach rapidly becomes computationally infeasible as the problem size increases.

This is not something that is particular for this problem, and is a major drawback of the solution of multistage stochastic optimization problems by dynamic programming in general: these dynamic programs often do not run in time that is bounded by a polynomial in the size of the input. One of the reasons behind the explosion in the running time is the corresponding explosion in the size of the

state space and/or action space. (In our case, we only had an explosion of the state space.)

In this paper, we will study the stochastic dynamic program introduced above. We will show that under some assumptions on the input, we can find, for any $\varepsilon > 0$, an approximation to the value of $V_0(s_0)$, such that the approximate value is at least $V_0(s_0)$ and is at most $(1+\varepsilon)V_0(s_0)$, and we can find this approximation in time that is bounded by a polynomial in the size of the input and $1/\varepsilon$. In other words, the method gives a fully polynomial-time approximation scheme (FPTAS) for this particular stochastic dynamic program.

Recently, Halman, Klabjan, Mostagir, Orlin and Simchi-Levi [20] proposed a method to address the explosion of the state and action space in a dynamic program to solve a class of problems they dub "Single-Item Stochastic Lot-Sizing Problems with Discrete Demand". Our paper is a generalization of their method — we will show how their methods can be adapted such that some of their assumptions imposed on the dynamic program can be relaxed (and hence applied to our setting for the American option pricing). More specifically, whereas Halman et al. [20] require that all functions involved are integer-valued functions, we only assume a bound on the reciprocal of the smallest nonzero function value (this will allow us to get an FPTAS, if this bound is polynomial in the size of the input).

On the other hand, we will also show that an extension to dynamic programs with even a 2-dimensional state space will not work in this framework.

There is a substantial literature on methods for the transformation of *deterministic* dynamic programs to yield corresponding FPTASs for broad class of discrete optimization problems. The unifying approach of Woeginger [28] provides a clear-eyed view of the extent to which this problem is very well understood in the

deterministic setting, and provides an extensive survey of references to applications where such methods were applied on a more *ad hoc* basis. We believe that this in an interesting step forward in the development of an analogous understanding of techniques for the efficient approximation of stochastic dynamic programming computations.

Option pricing has been studied in the context of getting approximations for a dynamic program. Chalasani, Jha and Saias [10] study a more restricted dynamic program, based on the binomial pricing model introduced by Cox, Ross and Rubinstein [12]. They give approximations for certain options, and show that a more general problem of pricing arbitrary path-dependent options (options, for which the payoff does not just depend on the value of the underlying asset, but also on the value that the underlying asset had over time) is $\#\mathcal{P}$-hard. We show that even in a rather special case of the option pricing problem that we consider (which only allows non-path-dependent option), the option pricing problem is $\#\mathcal{P}$-hard.

5.2 Approximating sets

In this section, we give the main combinatorial construction that underlies our approach, in which we compute a sparse representation of the domain of our function with sufficiently strong properties. As in Halman et al.[20], we will work with approximations of the value function at each step of the algorithm, instead of with the real function. The main idea is to use approximations based on a small (polynomial number) of points in the domain of the function. We will now make this precise for functions in general and then give an example of how this can be used to transform a dynamic program into a FPTAS in the next section.

Let $f : D \to R^{\geq 0}$ be a function that we want to approximate. We will assume

a total order on D, which is assumed to be finite, and also need the minimal data structures needed (or a compact representation) so that we can perform a bisection search over value in D. Consider a function $f : D \to \mathbb{R}^{\geq 0}$: then $\hat{f} : D \to \mathbb{R}^{\geq 0}$ is a *K-approximation* of f if $f(x) \leq \hat{f}(x) \leq Kf(x)$ for each $x \in D$. (We have followed the notation of [20] in calling this parameter K, but it might be more intuitive to this of K as being $1 + \epsilon$, for some fixed $\epsilon > 0$.)

The central definition of our approximation approach is the following definition, which is a somewhat more general structure than a similar one proposed by Halman et al.[20].

Definition 18. *Suppose $f : D \to \mathbb{R}^{\geq 0}$ is nondecreasing. Then $S \subseteq D$ is a K-approximating set for f, if*

(P1) $\min\{x \in D\} \in S$ *and* $\max\{x \in D\} \in S$

(P2) for $x, y \in S$ such that $y = \min\{z \in S, z > x\}$, we have that $f(y) \leq Kf(x)$ or $y = \min\{z \in D, z > x\}$ (or both).

Note how this definition differs from the notion of K-approximating sets in [20]:

1. We do not require that the "gaps" in S have a special size (half of property 4 in [20]);

2. By requiring that $\max\{x \in D\} \in S$, we get the third property in [20] for free.

Claim 19. *Let $f : D \to \mathbb{R}^{\geq 0}$ be nondecreasing. Let $S \subseteq D$ be a K-approximating set for f. Then for each $z \in D$, there exists $x \in S$ such that $z \leq x$ and $f(x) \leq Kf(z)$.*

Proof. If $z \in S$ then the claim obviously holds. Suppose $z \notin S$. By property (P2) of K-approximating sets, we have that if there are $x, y \in S$ such that $x < z < y$ then $f(y) \leq Kf(x)$. This would imply the claim: $f(x) \leq f(y) \leq Kf(x) \leq Kf(z)$ by monotonicity of f. By property (P1) of K-approximating sets, such x and y exist. □

Lemma 20. *Let $f : D \to \mathbb{R}^{\geq 0}$ be nondecreasing, and let $\mu = \min\{f(x) : f(x) > 0\}$. We can find a K-approximating set by evaluating f at $O(\log |D| \log(\max_x f(x)/\mu)/(K-1))$ points, and additional running time of the same order. Moreover, as this immediately implies, the same bound holds for the size of the approximating set.*

Proof. The main idea of the algorithm is as follows: if for any given point x we find the largest point y in D (possibly x itself) that evaluates to within a factor of K of the value of x, then the next point larger than y certainly evaluates to a value greater than that factor; this allows us to simultaneously find points that are (from their function value) sufficiently close together, without having to generate too many of them.

1. $x_0 \leftarrow \min\{y \in D\}$

   $S \leftarrow \{x_0\}$

   $i \leftarrow 0$

2. **while** $x_i < \max\{y \in D\}$ **do**

   (a) using bisection search, find the largest $y \in D$, such that $f(y) \leq Kf(x_i)$

   (b) $S \leftarrow S \cup \{y\}$

   (c) $i \leftarrow i + 1$

   (d) **if** $y < \max\{z \in D\}$ **then**

i. $x_i \leftarrow \min\{z : z \in D, z > y\}$

ii. $S \leftarrow S \cup \{x_i\}$

else $x_i \leftarrow y$.

Correctness: S is obviously a subset of D. The minimum of D is included in S at the start. Observe that for each iteration of the **while** loop, the value x_i computed in that iteration is in S by the end of that iteration. Hence, provided the algorithm terminates (which we will conclude below as part of the running time analysis), then the maximum element in D has been included S as well. Furthermore, note that since the point y found in Step 2(a) is at least x_i (since x_i satisfies the condition trivially), the points that the algorithm (potentially) adds to S are nondecreasing in D (that is, the points z for which we "update" $S \leftarrow S \cup z$ in Steps 2(b) and 2(d)). To prove property (P2), we therefore show that each new element added to S satisfies this property with respect to the previous new element added to S. Consider Step 2(b): this changes S only if $y \neq x_i$ (since as we remarked above, x_i is already in S), and in that case we know that $f(y) \leq Kf(x_i)$, and hence property (P2) is satisfied. Consider Step 2(d): in this case, we have that x_i is the next largest point than y, and Step 2(b) of that iteration of the **while** loop ensures that y is in S, and so again, property (P2) is satisfied.

Running time analysis: The bisection search takes at most $\log |D|$ steps. Note that with the possible exception of the last iteration, $f(x_{i+1}) > Kf(x_i)$ for each i. So the algorithm terminates, and the number of iterations is bounded by $1+\log_K(\max_x f(x)/\mu)$. Noting that $\log_K x = O(\log x/(K-1))$ gives the claim. □

Lemma 21 (basically Definition 5.4 and Proposition 5.5 in [20])**.** *Let* $f : D \to \mathbb{R}^{\geq 0}$

be nondecreasing, and let S be a K-approximating set of f. Let

$$\tilde{f}(x) = f(\min\{y \in S : y \geq x\}).$$

Then $\tilde{f}$ is a K-approximation of f and $\tilde{f}$ is nondecreasing.

Proof. $\tilde{f}(x) \geq f(x)$ by the monotonicity of f. If $x \in S$, then $\tilde{f}(x) = f(x)$, and the claim is obvious. If not, then $\tilde{f}(x) = f(\min\{y \in S : y \geq x\}) \leq Kf(\max\{y \in S : y \leq x\}) \leq Kf(x)$. □

Lemma 22 (basically Definition 3.6 and Proposition 3.7 in [20])**.** *Let $D \subseteq \mathbb{R}$. Let $f : D \to \mathbb{R}^{\geq 0}$ be nondecreasing* and convex*, and let S be a K-approximating set of f. Let $\ell(x) = \max\{y \in S : y \leq x\}$ and $r(x) = \min\{y \in S : y \geq x\}$. Let*

$$\hat{f}(x) = \frac{x - \ell(x)}{r(x) - \ell(x)} f(\ell(x)) + \frac{r(x) - x}{r(x) - \ell(x)} f(r(x))$$

when $\ell(x) \neq r(x)$, and $\hat{f}(x) = f(x)$ otherwise. Then $\hat{f}$ is a K-approximation of f. (Furthermore, $\hat{f}$ is nondecreasing and convex.)

Proof. If $x \in S$, then $\hat{f}(x) = f(x)$. Otherwise, convexity implies that $\hat{f}(x) \geq f(x)$. Finally, noting that $f(r(x)) \leq Kf(\ell(x))$, we see that $\hat{f}(x) \leq Kf(x)$. □

5.3 An application to pricing American options

We will show how the approximations developed in the previous section can yield a FPTAS for a stochastic dynamic program by giving an example of such a program and showing how to apply these ideas. The dynamic program finds the fair price of an American option with finite time horizon, where the option price depends on a Markov process.

Let $G_t(s)$ denote the payoff for the option, if it is exercised at time t, and the state of the system at time t is s. For ease of exposition we will assume that the

state is the price of the underlying asset (perhaps after a suitable transformation). The fact that we assume a finite time horizon means that there is some T such that $G_t(\cdot) = 0$ for $t > T$.

Writing S_t for (the stochastic variable representing) the stock price at t, we then get the following recursive definition for the fair price of the option at time t:

$$V_t(s_t) = \max\{G_t(s_t), E_{S_{t+1}}[V_{t+1}(S_{t+1})|S_t = s_t]\},$$

and, obviously, $V_{T+1}(\cdot) = 0$.

We make the assumption that the state space (for every t) is finite, and that the transition probabilities depend only on the difference between the stock price at subsequent times (if one assumes arbitrary transition probabilities, then the sheer size of the input is such that the dynamic program is already solvable in polynomial-time). We thus write

$$V_t(s_t) = \max\{G_t(s_t), \sum_{s_{t+1}} V_{t+1}(s_{t+1}) p^{(t)}_{s_{t+1}-s_t}\}$$

for some given functions $G_t(\cdot)$ and $\{p^{(t)}_\Delta\}_\Delta$. We are interested in computing $V_0(s_0)$ for some s_0.

Under certain assumptions on G_t we can now approximate $V_0(s_0)$ efficiently within $(1+\varepsilon)$ of the true value.

Proposition 23. *If G_t is monotone (nonincreasing or nondecreasing for all t) and nonnegative, and $\min\{G_t(x) : x \text{ such that } G_t(x) > 0\} > \pi_t$, where $\pi_t = \prod_{t'=t}^{(T)} \min_\Delta\{p^{(t')}_\Delta : p^{(t')}_\Delta > 0\}$, then there exists a FPTAS for calculating $V_0(s_0)$.*

Proof. We define $\bar{V}_t$ and $\tilde{V}_t$ recursively as follows:

$$\begin{aligned} \bar{V}_t(s_t) &= \max\{G_t(s_t), \sum_{s_{t+1}} \tilde{V}_{t+1}(s_{t+1}) p^{(t)}_{s_{t+1}-s_t}\} \\ &= \max\{G_t(s_t), \sum_{\Delta_{t+1}} \tilde{V}_{t+1}(s_t + \Delta_{t+1}) p^{(t)}_{\Delta_{t+1}}\}, \end{aligned}$$

and $\tilde{V}_t$ is the $(\tilde{f})$ approximation (as defined in Lemma 21) of $\bar{V}_t$ using a K-approximating set, and $\tilde{V}_{T+1} = V_{T+1} = 0$.

Claim 24. *$\bar{V}_t$ is nondecreasing (nonincreasing) and nonnegative if all G_t are nondecreasing (nonincreasing).*

Proof. By induction on t. $\bar{V}_T = G_T$, which is nondecreasing (nonincreasing) and nonnegative by assumption.

Assume the claim holds for $t+1$. Now $\tilde{V}_{t+1}$ is the $(\tilde{f})$ approximation of $\bar{V}_{t+1}$ using a K-approximating set, which is well-defined since $\bar{V}_{t+1}$ is monotone and nonnegative (where it is easy to see that the definitions in the previous section also extend to nonincreasing functions). Note that approximations inherit the monotonicity.

$\bar{V}_t$ is the maximum of two nondecreasing (nonincreasing) nonnegative functions, since the last part of its definition is a nonnegative combination of nondecreasing (nonincreasing) functions. □

Therefore $\tilde{V}_t$ and $\bar{V}_t$ are well defined.

Claim 25. *$\tilde{V}_t$ is a $K^{(T-t+1)}$-approximation of V_t.*

Proof. By induction on t. Note that $\bar{V}_T = V_T$, and so $\tilde{V}_T$ is a K-approximation of V_T.

Now suppose the claim for $t+1$. For any s we have

$$\begin{aligned}
\bar{V}_t(s) &= \max\{G_t(s), \sum_{\Delta_{t+1}} \tilde{V}_{t+1}(s+\Delta_{t+1}) p^{(t)}_{\Delta_{t+1}}\} \\
&\geq \max\{G_t(s), \sum_{\Delta_{t+1}} V_{t+1}(s+\Delta_{t+1}) p^{(t)}_{\Delta_{t+1}}\} \\
&= V_t(s)
\end{aligned}$$

and

$$\begin{aligned}
\bar{V}_t(s) &= \max\{G_t(s), \sum_{\Delta_{t+1}} \tilde{V}_{t+1}(s+\Delta_{t+1}) p^{(t)}_{\Delta_{t+1}}\} \\
&\leq \max\{G_t(s), K^{(T-t)} \sum_{\Delta_{t+1}} V_{t+1}(s+\Delta_{t+1}) p^{(t)}_{\Delta_{t+1}}\} \\
&\leq \max\{K^{(T-t)} G_t(s), K^{(T-t)} \sum_{\Delta_{t+1}} V_{t+1}(s+\Delta_{t+1}) p^{(t)}_{\Delta_{t+1}}\} \\
&\leq K^{(T-t)} V_t(s).
\end{aligned}$$

So $\bar{V}_t$ is a $K^{(T-t)}$-approximation to V_t, and since $\tilde{V}_t$ is a K-approximation to $\bar{V}_t$, we get that $\tilde{V}_t$ is a $K^{(T-t+1)}$-approximation of V_t. □

We can view the recursive definition above as an algorithm. If we choose $K = 1 + \varepsilon/(2(T+1))$ then $\tilde{V}_0(s_0)$ is a $(1+\varepsilon)$ approximation of $V_0(s_0)$ since $K^{(T+1)} = (1 + \varepsilon/(2(T+1)))^{(T+1)} \leq 1 + \varepsilon$ for ε small.

The only thing left to show is that the running time is polynomial in the size of the input and $1/\varepsilon$.

Claim 26. $\min\{\bar{V}_t(x) : \bar{V}_t(x) > 0\} > \pi_t$. *(Recall* $\pi_t = \prod_{t'=t}^{T} \min_\Delta\{p^{(t')}_\Delta : p^{(t')}_\Delta > 0\}$*.)*

Proof. By (backwards) induction on t. $\bar{V}_T(\cdot) = G_T(\cdot)$, so the claim holds by the assumption that $\min\{G_t(x) : x \text{ such that } G_t(x) > 0\} > \pi_t$.

Suppose now the claim is true for $t+1$. Since $\tilde{V}_{t+1}(\cdot)$ is a K-approximation of $\bar{V}_{t+1}(\cdot)$, $\tilde{V}_{t+1}(x) = 0$ iff $\bar{V}_{t+1}(x) = 0$. Also, $\tilde{V}_{t+1}(\cdot) \geq \bar{V}_{t+1}(\cdot)$. Therefore $\min\{\tilde{V}_{t+1}(x) : \tilde{V}_{t+1}(x) > 0\} = \min\{\tilde{V}_{t+1}(x) : \bar{V}_{t+1}(x) > 0\} \geq \min\{\bar{V}_{t+1}(x) : \bar{V}_{t+1}(x) > 0\} > \pi_{t+1}$, where the last inequality follows from the inductive hypothesis.

Now $\bar{V}_t(x) = \max\{G_t(x), \sum_{\Delta_{t+1}} \tilde{V}_{t+1}(x+\Delta_{t+1}) p^{(t)}_{\Delta_{t+1}}\}$, and so if $\bar{V}_t(x) > 0$, then either $\bar{V}_t(x) = G_t(x)$ such that $G_t(x) > 0$, which by the assumption on $G_t(\cdot)$

implies $\bar{V}_t(x) > \pi_t$, or $\bar{V}_t(x) = \sum_{\Delta_{t+1}} \tilde{V}_{t+1}(x + \Delta_{t+1}) p^{(t)}_{\Delta_{t+1}} > 0$, which implies $\bar{V}_t(x) \geq \min_\Delta \{p^{(t)}_\Delta : p^{(t)}_\Delta > 0\} \tilde{V}_{t+1}(y)$ for some y such that $\tilde{V}_{t+1}(y) > 0$. By our previous argument we know that for such y we have $\tilde{V}_{t+1}(y) > \pi_{t+1}$. Combining the inequalities, we see that $\bar{V}_t(x) > \min_\Delta \{p^{(t)}_\Delta : p^{(t)}_\Delta > 0\} \pi_{t+1} = \pi_t$ if $\bar{V}_t(x) > 0$. □

Also note that $\bar{V}_t$ is bounded from above by $\max_{t,x} G_t(x)$. We shall call this quantity U.

By Lemma 20, we can conclude that the K-approximating set for $\bar{V}_t$ has size $O(\log |D| \log(U/\pi_t)/\ (K-1))$.

For each of the points that are queried when creating the K-approximating set of $\bar{V}_t$ (of which there are $O(\log |D| \log(U/\pi_t)/(K-1))$ by Lemma 20), one evaluation of G_t was needed, and N_t evaluations of $\tilde{V}_{t+1}$, where N_t is the number of positive transition probabilities for time $t+1$ (i.e., $N_t = \#\{p^{(t)}_\Delta > 0\}$). An evaluation of $\tilde{V}_{t+1}$ can be done in $O(\log(\log |D| \log(U/\pi_{t+1})/(K-1)))$ time, since it takes $\log m$ steps to find the closest of m points, using bisection search.

We therefore conclude that for each t we need $O(\log |D| \log(U/\pi_t)/(K-1) N_t \log(\log |D| \log(U/\ \pi_{t+1})/(K-1)))$ time. Being generous, we get that the total running time is bounded by $O(T N_{\max} \log |D| \log(U/\pi_1)/(K-1)(\log\log |D| + \log\log(U/\pi_1) + \log(1/(K-1)))) = O(2(1/\varepsilon) T(T+1) N_{\max} \log |D| \ \log(U/\pi_1)(\log\log |D| + \log\log(U/\pi_1) + \log(2(T+1)/\varepsilon)))$, where $N_{\max} = \max_t N_t$. We conclude that the running time is bounded by a polynomial in the size of the input and $1/\varepsilon$.

This concludes the proof of the proposition. □

By a similar argument, we get the following result:

Proposition 27. *Let G_t be convex and nonnegative for all t, and let* $\min\{G_t(x) :$

x such that $G_t(x) > 0\} > \pi_t$, where $\pi_t = \prod_{t'=t}^{T} \min_\Delta\{p_\Delta^{(t')} : p_\Delta^{(t')} > 0\}$, then there exists a FPTAS for calculating $V_0(s_0)$.

Now we need the other approximation ($\hat{f}$ as defined in Lemma 22) (where the fact that we can find K-approximating sets for these functions is not hard to see — we can find the maximum of a convex function in polynomial-time and can then split the function up into two parts which are convex monotone functions, as was analogously done in [20]). The proof follows the previous one in a straightforward manner, and is therefore omitted (the proof that $V_t(\cdot)$ is convex for all t is easy: it is the maximum of a convex function and the sum of convex functions. Note that for this to work, it is essential to work with the $\hat{f}$ approximation as defined in Lemma 22, as this approximation preserves convexity).

5.4 The $\#\mathcal{P}$-hardness of the American option pricing problem

Lemma 28. *Finding $V_0(s_0)$ as defined above and under the assumptions stated, is $\#\mathcal{P}$-hard.*

Proof. Our proof is analogous to the $\#\mathcal{P}$-hardness proof of the single-item stochastic lot sizing problem of Halman et al. [20].

We will use the K-th largest subset problem as the basis for our proof. It is well known (but apparently only a folklore result) that this problem is $\#\mathcal{P}$-hard.It thus suffices to reduce the decision problem of the K-th largest subset problem to the option pricing problem, with the decision "is the option worth more than x?".

In Garey & Johnson [16], the following definition of K-th largest subset is given:

INSTANCE: Finite set A, size $s(a) \in \mathbb{Z}^{\geq 0}$ for each $a \in A$, positive integers K and B

QUESTION: Are there K or more distinct subsets $A' \subseteq A$ for which the sum of the sizes of the elements in A' does not exceed B?

Given an instance of the K-th largest subset problem, where without loss of generality we assume $A = \{1, 2, \ldots, T\}$, create the following instance to the option pricing problem:

$$\begin{aligned} S_0 &= 0 \\ S_t &= \begin{cases} S_{t-1} + s(t) & \text{with probability } 1/2 \\ S_{t-1} & \text{with probability } 1/2 \end{cases} \\ G_T(s) &= \begin{cases} 1 & \text{if } s \leq B \\ 0 & \text{if } s > B \end{cases} \\ G_t(s) &= 0 \text{ for all } t < T \text{ and for all } s. \end{aligned}$$

Note that $G_t(\cdot)$ obviously satisfies the assumptions of both Proposition 23 and Proposition 27.

Obviously it is never profitable to exercise the option, except at time T. The price of the option at time 0 is just the expected payoff, which is exactly equal to the probability that the stock price at time T is at most B. Noting that $P(S_T \leq B) = (\#$ of distinct subsets of A for which the sum of the sizes of the elements does not exceed $B\ )/2^T$ concludes the reduction. □

5.5 Further extensions

It is not hard to extend the notions of approximating sets in various ways. They can be extended to general unimodal functions (see Halman et al. [20]). To be

able to find a K-approximating set efficiently in this case, one needs to be able to find the optimum of the function efficiently. Assuming convexity (or concavity, with slight modifications to the definitions above) suffices. However, for general unimodal functions it is not possible to find the optimum efficiently (take the function $f(x) = c_1$ for all $x \neq x_0$, and $f(x_0) = c_0$).

A natural generalization in the case of option pricing would be to consider options on a basket of assets. However, a generalization of this approach to multivariate functions does not seem to work, even when assuming convexity.

The idea of an approximating set is that we can estimate the function value of any point not in the approximating set, within K times its real value, and never underestimate the value (conforming the definition of K-approximation above), using only the function value of the points in the approximating set.

We will now show that there exist (convex, monotone in both coordinates) functions for which such sets need $\Omega(n)$ points, where the domain is of the function is $\{0, 1, \ldots, n\}^2$. We will actually show a slightly stronger statement: we drop the restriction that we may not underestimate the function, and let "relaxed K-approximating set" be any set such that the function value at any point can be estimated within a factor K, using only the function values at the points in the approximating set.

Claim 29. *For the following function defined on* $\{0, 1, \ldots, n\}^2$*:*

$$f(x_1, x_2) = \begin{cases} 1 & \text{for } x_1 + x_2 \leq n \\ K^3 & \text{for } x_1 + x_2 = n + 1 \\ 2(x_1 + x_2 - n - 1)K^3 & \text{otherwise} \end{cases}$$

any "relaxed K-approximating set", such that the function value of any point not in the set can be estimated within a factor of K, needs $\Omega(n)$ points.

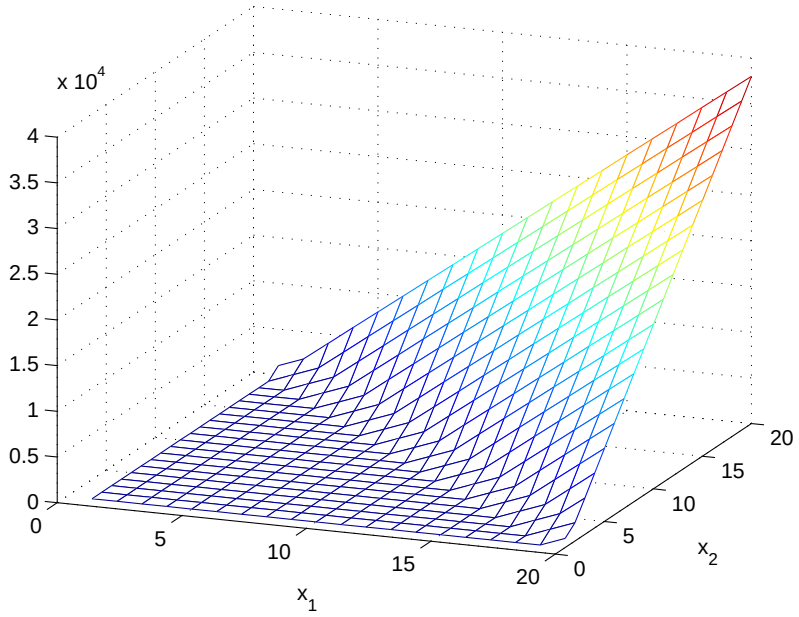


Figure 5.1: Function of Claim 29 for $n = 20$ and $K = 10$

Proof. Note that f is indeed increasing in both coordinates and convex.

Suppose now, by contradiction, that there does exist a relaxed K-approximating set of size $o(n)$. That means that there exists a relaxed K-approximating set that does not contain all the points such that $x_1 + x_2 = n + 1$. Let $(\bar{x}_1, \bar{x}_2)$ be such a point that is not included in the K-approximating set.

We will now show that even with the knowledge of the function values of all points except $(\bar{x}_1, \bar{x}_2)$, we still cannot estimate $f(\bar{x}_1, \bar{x}_2)$ within a factor K.

First of all, monotonicity just gives $1 \leq f(\bar{x}_1, \bar{x}_2) \leq 2K^3$.

For the best upper bound that can be found using convexity we look at the

following LP:

$$\begin{aligned}
\text{minimize} \quad & \sum_i \lambda_i f(x^{(i)}) & \\
\text{s.t.} \quad & \sum_i \lambda_i = 1 & \text{all } i \\
& \sum_i \lambda_i x_1^{(i)} = \bar{x}_1 & \\
& \sum_i \lambda_i x_2^{(i)} = \bar{x}_2 & \\
& \lambda_i \geq 0 &
\end{aligned}$$

where the $x^{(i)}$s are an enumeration of $\{0, 1, \ldots, n\}^2$, and its dual

$$\begin{aligned}
\text{maximize} \quad & \alpha + \bar{x}_1\beta_1 + \bar{x}_2\beta_2 & \\
\text{s.t.} \quad & \alpha + x_1^{(i)}\beta_1 + x_2^{(i)}\beta_2 \leq f(x_1^{(i)}, x_2^{(i)}) & \text{all } i \\
& \alpha, \beta_1, \beta_2 \in \mathbb{R}. &
\end{aligned}$$

It is easily verified that $\lambda_1 = \lambda_2 = \frac{1}{2}$, with $(x_1^{(1)}, x_2^{(1)}) = (\bar{x}_1 - 1, \bar{x}_2 + 1)$ and $(x_1^{(2)}, x_2^{(2)}) = (\bar{x}_1 - 1, \bar{x}_2 + 1)$, is a feasible primal solution with objective K^3. It is not much harder to verify that $\alpha = 1 - n(K^3 - 1)$, $\beta_1 = \beta_2 = K^3 - 1$ is a feasible dual solution with the same objective value. By weak duality we conclude that these must be optimal solutions.

So, we conclude that monotonicity and convexity gives only enough information to conclude that $f(\bar{x}) \in [1, K^3]$, contradicting the fact that we have a K-approximating set. □

Corollary 30. *The number of points in any "(relaxed) K-approximating set" for f defined above cannot be polynomially bounded in terms of $O(\log |D|)$ and $O(\log(\max_x f(x)/(\min\{f(x) : f(x) > 0\})))$.*

Appendix A

Tomato Data Set

A.1 Markers

The table on the next page shows which markers were used for each chromosome, and the (probabilistic) position of the marker (in cM).

	marker	position		marker	position		marker	position
1	TG58	14	2	LEC7P21	0	3	Py1	5
	SSR51	40		SSR40	22		TG585	40
	SSR134	47		SSR356	44		LPT2E21	61
	T1409	77		SSR349	48		SSR111	76
	SSR222	98		SSR605	49		SSR22	109
	CT267	114		T0562	71		T0794	120
	SSR308	121		SSR26	77		SSR320	158
	SSR117	138		FW2.2	90		SSR11	164
	SSR65	158		T1480	106		SSR27	169
				T0634	130			

	marker	position		marker	position		marker	position
4	T707	0	5	LEX13I3	13	6	T892	14
	SSR310	15		SSR115	35		T507	25
	T1405	77		LEX13G5	79		SSR128	35
	SSR146	102		TG23	99		SSR578	44
	SSR188	136		T633	119		FTG275	70
							TG279	80
							SSR350	100

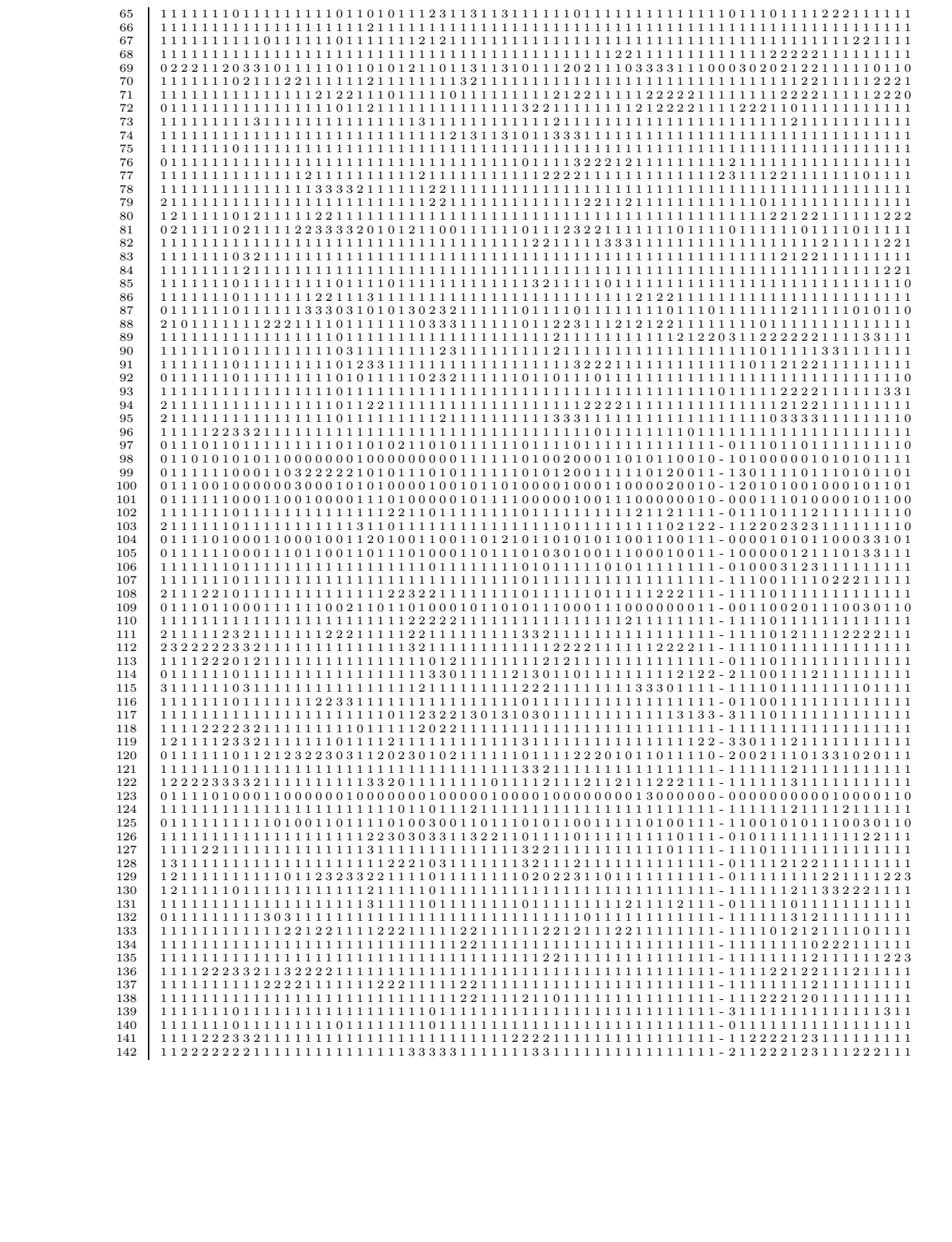
	marker	position		marker	position		marker	position
7	SSR241	0	8	SSR344	1	9	TG18	14
	SSR52	4		SSR244	7		T1617	32
	I3	43		SSR327	23		SSR70	42
	SSR45	58		TG302	37		SSR383	58
	T1738	73		SR38	55		T1190	77
	CT114	96		SSR594	55		T1519	90
				T1359	73		T1065	116
				CT68	87			

	marker	position		marker	position		marker	position
10	TG230	0	11	TG497	1	12	TG180	9
	TG303	11		SSR80	20		TG68	20
	SSR34	28		SSR67	24		LET8K4	41
	SSR218	34		SSR46	40		SSR44	60
	SSR318	35		TG400	57		T801	70
	SSR248	36		LEC24C3	76		T1305	90
	SSR85	55		T302	90		T800	113
	T1682	66		TG393	103			
	SSR223	87						

A.2 Base Set Plants

Here is a summary of the marker of the plants in the base set. Each row represents a plant, the plant number is given in the first column. The markers are listed in order of the chromosome they are on, and within the same chromosome in order of their position. 0 indicates inconclusive data from the marker lab, 1 means homozygous domesticated, 2 heterozygous, 3 homozygous wild, and finally "-" indicates missing data.

2	1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
3	0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
4	1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 2 2 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1
5	1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 3 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 0 1 1 0 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 0 1 1 1 1
6	1 1 0 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1
7	1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 3 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1
8	2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0
9	1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
10	1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2
11	1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1
12	1 1 1 1 1 3 2 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 0 1 1 1 1 1 1 1 1 1 0 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 0
13	1 1 1 2 3 2 2 0 3 2 1 1 1 1 1 1 1 0 1 1 0 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 0 1 1 1 1 1 1 1 1 0 1 1 2 2 2 1 3 3 1 1 1 1 2 2 1 1 1
14	1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 0 1 1 1 1 2 1 1 1 1 1 1 1 0 1 1 2 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 3 3 3 3 1 1 1 1 1 1 1 1 0
15	1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 0 1 1 1 1 0 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 0 1 1 1 1 2 2 2 0 1 0 1 1 0
16	1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 1 1 1 1 1 1 1 0 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0
17	1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 2 1 1 1 1 1 1 1 1 1 1 1
18	1 1 1 1 1 1 1 1 3 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
19	1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 0 1 1 1 1 0 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 0 1 1 1 1 3 1 1 1 1 1 2 1 1
20	1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 0 1 1 1 1 2 1 2 2 1 1 1 1 1 1 1 1 1
21	1 1 1 2 3 2 2 0 2 3 1 0 1 1 1 1 1 0 1 1 1 1 0 3 2 0 1 0 2 1 1 1 1 3 1 0 1 1 1 0 0 1 1 1 1 1 1 1 1 1 1 1 0 1 0 1 0 0 1 1 1 1 1 1 1 1 1 1 1 0 3 1 0
22	1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 0 1 1 2 2 1 1 1 1 1 1 1 2 2 2 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
23	1 1 1 1 1 0 2 3 3 2 2 1 1 3 2 2 2 2 1 1 0 1 0 1 1 1 0 1 1 1 1 1 1 1 1 0 1 0 1 1 0 0 1 1 1 0 1 1 0 0 1 1 0 0 0 1 1 0 1 1 1 1 1 1 1 1 1 2 3 1 2 0 3
24	2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 2
25	1 1 1 1 1 1 1 0 3 0 1 0 1 1 1 1 1 1 1 1 0 1 1 1 1 0 1 1 0 1 1 1 2 1 1 0 1 1 1 1 0 1 1 1 1 1 1 1 1 1 0 1 1 0 1 1 1 1 1 1 1 2 1 1 1 1 0 1 2 0 3 3 3
26	1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 2 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1
27	1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 3 1 1 1 1 1 1 2 2 2
28	1 1 1 1 1 1 1 0 3 3 1 1 1 1 1 1 1 0 1 1 2 2 1 1 0 1 1 1 1 1 1 1 1 3 1 0 2 2 1 1 1 1 1 1 1 1 1 1 1 0 0 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 3 3 3
29	1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 0 1 1 1 2 1 1 1 1 1 1 1 1 1
30	2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2
31	1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1
32	1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 2 2 1 1 1 1 1 1 1 1 1 1 1 3 1 0 1 1 1 1 1 1 1 1 1 3 1 1 1 1 1 1 1 1 1 0 1 1 2 1 1 1 1 3 1 1 2 3 2 2 2 2 2
33	1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 3 3 3 1 1 1 1 1 1 1 2 1 1 1
34	1 1 1 1 3 2 2 3 3 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 2 2 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 3 3 3 3 1 1 1 1 1 1 1 3 1 1 1
35	1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 3 3 0 1 1 1 1 1 1 1 1 1 1 1 1 3 1 1 1 1 1 1 0 3 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 3 3 2 3 3 3 1 1 1 1 1 1 1 1 1 1 1
36	1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 2 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
37	1 1 1 1 2 1 2 3 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1
38	1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 2 2 2 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 3 1 1 1 1 1 1 1 1 1
39	0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 3 1 1 1 1 1 1 1 1 1 1 1 3 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0
40	1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 2 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 3 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 1 1 1 1 1 1 1 1 1 1
41	1 1 1 1 2 3 3 3 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 3 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 2 1 1 1 1 1 1 1 1 1 1 1 1
42	1 2 1 1 1 1 2 3 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 1 1 1
43	1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 3 0 0 3 3 3 1 1 1 2 2 2 1 1 1
44	1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 2 1 1 1 2 1 1 1 1 1 1 1 1 1
45	1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 3 2 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1
46	3 1 1 1 1 1 1 0 1 1 1 1 1 1 1 2 2 2 1 1 3 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 3 1 0 1 1 1 1 1 3 3 3 3 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1
47	2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 3 1 1 1 1 2 2 2 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 3 1 1 1 1 1 1 1 1 1
48	2 1 1 1 0 1 1 1 1 1 1 1 1 1 1 0 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
49	1 1 1 1 1 1 1 0 1 1 1 0 1 1 1 2 1 1 3 2 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 0 1 1 1 1 1 1 1 0 1 1 2 1 2 2 1 1 1 1 1 0 1 1 1 1 1 3 1 1 1 1 0 1 1 1 1 1 1
50	1 1 1 1 1 1 1 0 1 1 1 0 1 1 1 1 1 2 3 2 1 1 2 2 2 1 1 1 2 1 1 1 1 1 1 0 1 1 1 1 0 1 0 1 1 2 1 1 1 1 2 2 2 2 0 3 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1
51	1 1 1 1 1 1 1 0 1 1 1 1 2 1 1 1 1 1 1 1 0 1 0 1 0 0 1 1 1 1 0 1 1 1 1 0 1 1 1 1 1 1 0 1 1 1 1 1 1 1 2 1 1 1 1 0 1 1 1 0 0 3 3 3 1 1 2 1 0 1 1 1 1
52	1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 0 2 2 1 0 0 1 1 1 1 1 1 1 1 1 0 1 1 1 1 0 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1
53	2 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 0 1 1 1 1 1 1 1 1 2 1 2 2 1 1 1 3 3 3 1 1 1
54	0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 2 2 1 2 2 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
55	1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 3 2 0 1 1 1 0 1 1 0 1 1 0 1 1 1 1 0 1 1 1 1 0 1 1 1 1 0 1 1 1 1 0 1 1 1 1 0 1 1 1 1 2 1 2 2 1 1 1 1 1 1 1 1 1
56	1 1 1 1 1 1 1 0 0 1 2 1 3 1 1 1 1 1 3 2 2 1 1 1 0 1 2 3 2 1 1 1 1 1 1 3 2 1 0 1 0 1 1 1 1 1 1 0 1 1 0 1 1 1 1 0 1 0 0 0 3 1 3 3 1 1 1 1 1 1 1 1 1
57	1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 3 2 0 1 0 1 2 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 0 1 0 1 1 0 1 1 1 1 0 1 1 1 1 1 1 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1
58	0 1 1 1 1 1 1 0 1 0 1 1 1 1 1 1 1 0 1 0 1 0 2 2 2 1 1 1 1 1 1 1 1 1 1 0 3 3 0 3 3 3 1 1 1 1 1 1 1 1 0 1 1 1 1 0 1 0 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1
59	1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 0 2 1 2 2 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1
60	1 1 1 1 1 1 1 0 1 1 2 2 3 1 1 1 1 1 1 1 1 1 3 3 3 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 3 3 1 1 0 1 1 1 1
61	1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
62	1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 3 3 3 3 3 1 1 1 1 1 1 1 1 1
63	1 1 1 1 0 1 1 0 1 1 1 1 1 1 1 1 1 0 1 1 0 1 0 1 1 1 1 3 1 1 1 1 1 1 1 0 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 1 1 1 0 1 1 1 1 1 1 1 1 1 1 3 3 3
64	1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 2 1 0 1 1 1 1 1 1 1 3 2 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1



BIBLIOGRAPHY

[1] Yair Bartal. Probabilistic approximation of metric spaces and its algorithmic applications. In *37th Annual Symposium on Foundations of Computer Science (Burlington, VT, 1996)*, pages 184–193. IEEE Comput. Soc. Press, Los Alamitos, CA, 1996.

[2] J. J. Bartholdi, III and L. K. Platzman. An $O(N \log N)$ planar travelling salesman heuristic based on spacefilling curves. *Oper. Res. Lett.*, 1(4):121–125, 1981/82.

[3] John J. Bartholdi, III, Loren K. Platzman, R. Lee Collins, and William H. Warden, III. A minimal technology routing system for meals on wheels. *Interfaces*, 13(3):1–8, 1984.

[4] S. Ben-David, A. Borodin, R. Karp, G. Tardos, and A. Wigderson. On the power of randomization in on-line algorithms. *Algorithmica*, 11(1):2–14, 1994.

[5] Dimitris Bertsimas. *Probabilistic Combinatorial Optimization Problems*. PhD thesis, MIT, Cambridge, Mass., 1988.

[6] Dimitris Bertsimas and Michelangelo Grigni. Worst-case examples for the spacefilling curve heuristic for the euclidean traveling salesman problem. *Operations Research Letters*, 8(5):241–244, October 1989.

[7] Dimitris J. Bertsimas, Patrick Jaillet, and Amedeo R. Odoni. A priori optimization. *Operations Research*, 38(6):1019–1033, 1990.

[8] A. Borodin and R. El-Yaniv. *Online Computation and Competitive Analysis*. Cambridge University Press, 1998.

[9] Allan Borodin, Morten N. Nielsen, and Charles Rackoff. (incremental) priority algorithms. In *SODA '02: Proceedings of the thirteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 752–761, Philadelphia, PA, USA, 2002. Society for Industrial and Applied Mathematics.

[10] Prasad Chalasani, Somesh Jha, and Isaac Saias. Approximate option pricing. *Algorithmica*, 25:2–21, 1999.

[11] George Christopher, Martin Farach, and Michael Trick. The structure of circular decomposable metrics. In *Algorithms—ESA '96 (Barcelona)*, volume 1136 of *Lecture Notes in Comput. Sci.*, pages 486–500. Springer, Berlin, 1996.

[12] John Cox, Stephen Ross, and Mark Rubinstein. Option pricing: a simplified approach. *Journal of Financial Economics*, 7:229–263, 1979.

[13] Vladimir G. Deĭneko, Rüdiger Rudolf, and Gerhard J. Woeginger. Sometimes travelling is easy: the master tour problem. *SIAM J. Discrete Math.*, 11(1):81–93, 1998.

[14] Jittat Fakcharoenphol, Satish Rao, and Kunal Talwar. A tight bound on approximating arbitrary metrics by tree metrics. In *Proceedings of the Thirty-Fifth Annual ACM Symposium on Theory of Computing*, pages 448–455, New York, 2003. ACM.

[15] Jittat Fakcharoenphol, Satish Rao, and Kunal Talwar. A tight bound on approximating arbitrary metrics by tree metrics. *J. Comput. System Sci.*, 69(3):485–497, 2004.

[16] Michael R. Garey and David S. Johnson. *Computers and intractability: A guide to the theory of NP-completeness*. W. H. Freeman and Co., San Francisco, CA, 1979.

[17] Anupam Gupta, Mohammad T. Hajiaghayi, and Harald Räcke. Oblivious network design. In *SODA '06: Proceedings of the seventeenth annual ACM-SIAM symposium on Discrete algorithm*, pages 970–979, New York, NY, USA, 2006. ACM Press.

[18] Mohammad T. Hajiaghayi, Robert Kleinberg, and Tom Leighton. Improved lower and upper bounds for universal tsp in planar metrics. In *SODA '06: Proceedings of the Seventeenth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 649–658, New York, NY, USA, 2006. ACM Press.

[19] J.B.S. Haldane. The combination of linkage values, and the calculation of distances between the loci of linked factors. *J. Geneti.*, 8:299–309.

[20] Nir Halman, Diego Klabjan, Mohamed Mostagir, James B. Orlin, and David Simchi-Levi. A fully polynomial time approximation scheme for single-item stochastic lot-sizing problems with discrete demand. MIT Sloan Research Paper No. 4582-06. A slightly later version available at: `https://netfiles.uiuc.edu/klabjan/www/articles/DSLS.pdf`, January 2006.

[21] Patrick Jaillet. Probabilistic traveling salesman problems. Technical Report 185, Operations Research Center, MIT, 1985.

[22] Patrick Jaillet. A priori solution of a traveling salesman problem in which a random subset of the customers are visited. *Operations Research*, 36:929–936, 1988.

[23] Lujun Jia, Guolong Lin, Guevara Noubir, Rajmohan Rajaraman, and Ravi Sundaram. Universal approximations for TSP, Steiner tree, and set cover. In *STOC '05: Proceedings of the Thirty-seventh Annual ACM Symposium on Theory of Computing*, pages 386–395, New York, NY, USA, 2005. ACM Press.

[24] Tom Leighton and Satish Rao. Multicommodity max-flow min-cut theorems and their use in designing approximation algorithms. *J. ACM*, 46(6):787–832, 1999.

[25] Aranyak Mehta, Amin Saberi, Umesh Vazirani, and Vijay Vazirani. Adwords and generalized on-line matching. In *FOCS '05: Proceedings of the 46th Annual IEEE Symposium on Foundations of Computer Science*, pages 264–273, Washington, DC, USA, 2005. IEEE Computer Society.

[26] Loren K. Platzman and III John J. Bartholdi. Spacefilling curves and the planar travelling salesman problem. *J. ACM*, 36(4):719–737, 1989.

[27] Henk C. Tijms. *Stochastic models*. Wiley Series in Probability and Mathematical Statistics: Applied Probability and Statistics. John Wiley & Sons Ltd., Chichester, 1994. An algorithmic approach.

[28] Gerhard J. Woeginger. When does a dynamic programming formulation guarantee the existence of an FPTAS? In *SODA '99: Proceedings of the Tenth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 820–829, Philadelphia, PA, USA, 1999. Society for Industrial and Applied Mathematics.