ABSTRACT

JACKSON, LAURA ELIZABETH. The Directional p-Median Problem with Applications to Traffic Quantization and Multiprocessor Scheduling.
(Under the direction of George N. Rouskas and Matthias F.M. Stallmann.)

An instance of a p-median problem gives n demand points. The objective is to locate p supply points in order to minimize the total distance of the demand points to their nearest supply point. P-median is polynomially solvable in one dimension but NP-hard in two or more dimensions, when either the Euclidean or the rectilinear distance measure is used. In this thesis, we treat the p-median problem under a new distance measure, the directional rectilinear distance, which requires the nearest supply point for a given demand point to lie above and to the right of it. This restriction has applications to multiprocessor scheduling of periodic tasks as well as to traffic quantization and Quality of Service scheduling in packet-switched computer networks. We show that the directional p-median problem is polynomially solvable in one dimension and give two algorithms. We prove the problem NP-hard in two or more dimensions and then present an efficient heuristic to solve it. Compared to the robust Teitz & Bart heuristic for p-median, our heuristic enjoys substantial speedup while sacrificing little in terms of solution quality, making it an ideal choice for our target applications with thousands of demand points.

The Directional p-Median Problem with Applications

to Traffic Quantization and Multiprocessor Scheduling

by

Laura Elizabeth Jackson

B.S. Mathematical Economics, M.S. Applied Science

A dissertation submitted to the Graduate Faculty of
North Carolina State University
in partial satisfaction of the
requirements for the degree of
Doctor of Philosophy

Department of Computer Science

Raleigh

2003

Approved By:

Dr. Carla Savage	Dr. Jeffrey Joines
Dr. George N. Rouskas Co-Chair of Advisory Committee	Dr. Matthias F.M. Stallmann Co-Chair of Advisory Committee

To my family — Mom, Dad, Jeff, and Eliza.

Biography

Laura Elizabeth Jackson was born in Greenwood, South Carolina, and moved to North Carolina at the age of nine. She graduated from high school in 1990 from the NC School of Science and Math in Durham. She was a James Monroe Scholar at the College of William and Mary in Williamsburg, Virginia, where she earned the BS in Mathematical Economics (1995, magna cum laude) and the MS in Operations Research (1997). Along the way, she spent one year learning German in Berlin and another year working as an actuary in Philadephia, Pennsylvania. She entered the NC State University operations research program in 1997 and served one year as president of its graduate student association. She later transferred to the computer science department and was named a GAANN (Graduate Assistance in Areas of National Need) Computational Science fellow. While in graduate school, she worked two years part-time in the Advanced Network Research group at MCNC, where she helped design the architecture and protocols for an all-optical local area computer network. Her publications have appeared in IEEE Computer magazine, IEEE Transactions on Parallel and Distributed Systems, and three conference proceedings. After graduation, she plans to pursue a career in university research and teaching.

Acknowledgements

I would like to thank the following great teachers for their dedication to excellence and for the example they have given me. At East Iredell Elementary school, algebra teacher Greg "Da Boss" Crowley captivated a room full of 13-year-olds with his energy and enthusiasm. At the North Carolina School of Science and Math, John Goebel's creative descriptions enabled me to "see" calculus. Larry Rabinowitz's probability course was the most challenging and rewarding class of my undergraduate career at William and Mary. In my first programming course, Rance Necaise's patient explanations led to a "Eureka!" moment when I first comprehended pointers. In Steve Park's two simulation courses, I learned when and when not to simulate, I learned to think before programming, and I learned how to verify code; in short, I learned how to do it right the first time. At NC State, Carla Savage introduced me to a whole new world of abstraction called graph theory. Matt Stallmann helped me hone my proof-writing skills. Finally, I would like to thank my advisor, George Rouskas. Through his expert guidance and encouragement I learned how to do research.

Contents

List of Figures **viii**

List of Tables **xiv**

Chapter

1 The p-median problem 1

1.1 A brief history 1

1.2 The p-median problem: continuous and discrete space 2

1.3 A new notion of distance: The directional distance metric 5

1.4 Organization of thesis 6

2 Applications of the directional p-median problem 8

2.1 Traffic quantization 8

2.2 Multiprocessor scheduling of periodic tasks 10

3 The directional p-median problem on the real line 13

3.1 Statement of the problem: DPM1 14

3.2 A dynamic programming algorithm for DPM1: Quantize 16

3.3 Performance penalty due to quantization for Quantize 21
3.4 Graph representation of DPM1: A faster algorithm 36
3.5 The stochastic directional p-median problem on the real line . . 39
3.5.1 Statement of the problem: SDPM1 39
3.5.2 Optimal solution through nonlinear programming 42
3.5.3 An approximate solution: Algorithm Quantize-Continuous 46
3.5.4 Performance of Quantize-Continuous on six input distributions 48
3.6 Application: Improved complexity for scheduling periodic tasks on a multiprocessor 54
3.7 Conclusion 56

4 The complexity of directional p-median in two or more dimensions 57
4.1 Directional p-median in two dimensions 59
4.2 Construction of the circuit for each variable 62
4.3 Alignment of clause points with variable circuits 64
4.4 Sketch of proof 68
4.5 Proof details 70
4.5.1 Notation 70
4.5.2 An equal number of SW and NE corners 70
4.5.3 Notation revisited: A point numbering scheme 73
4.5.4 A circuit lemma: The true partition and the false partition are the only optimal partitions 75
4.5.5 Two clause configuration lemmas: Only legitimate subsets may be in an optimal partition 77

5 An efficient heuristic algorithm for DPM2 84

5.1 Effect of distance characteristics on computational effort 84

5.2 Teitz & Bart vertex substitution heuristic for p-median 85

5.3 Heuristic concentration 88

5.4 The Teitz & Bart Restricted heuristic for DPM2 89

5.5 Evaluation of the Teitz & Bart Restricted heuristic 90

5.6 Conclusion 95

6 Summary and future work 110

6.1 Summary 110

6.2 Future directions 111

Bibliography 113

Appendix

A Algorithm Quantize for DPM1 116

B Algorithm Quantize-Continuous for SDPM1 118

C Algorithm Q-PD^2 for scheduling a quantized task set 120

List of Figures

Figure

1.1 Sample p-median problem instances in the plane, with $n = 9$ demand points and $p = 3$ supply points. 5

1.2 Demand points arranged in a downward slope yield the maximum number of directional intersection points. 7

3.1 Sample mapping of task densities to service levels. 14

3.2 Probability density functions for the uniform and triangle input distributions. 23

3.3 Probability density functions for the increasing and decreasing input distributions. 24

3.4 Probability density functions for the unimodal and bimodal input distributions. 25

3.5 Level curves in n of the normalized quantization load vs. number of supply points (p). Top: Uniform. Bottom: Triangle. 26

3.6 Level curves in n of the normalized quantization load vs. number of supply points (p). Top: Increasing. Bottom: Decreasing. . . . 27

3.7 Level curves in n of the normalized quantization load vs. number of supply points (p). Top: Unimodal. Bottom: Bimodal. 28

3.8 Uniform input: Level curves in p of the normalized quantization load, for 100 individual demand sets. Top: $n = 100$. Bottom: $n = 1000$. 30

3.9 Triangle input: Level curves in p of the normalized quantization load, for 100 individual task sets. Top: $n = 100$. Bottom: $n = 1000$. 31

3.10 Increasing input: Level curves in p of the normalized quantization load, for 100 individual demand sets. Top: $n = 100$. Bottom: $n = 1000$. 32

3.11 Decreasing input: Level curves in p of the normalized quantization load, for 100 individual demand sets. Top: $n = 100$. Bottom: $n = 1000$. 33

3.12 Unimodal input: Level curves in p of the normalized quantization load, for 100 individual demand sets. Top: $n = 100$. Bottom: $n = 1000$. 34

3.13 Bimodal input: Level curves in p of the normalized quantization load, for 100 individual demand sets. Top: $n = 100$. Bottom: $n = 1000$. 35

3.14 Graph representation of an instance of DPM1 with $n = 5$. . . . 36

3.15 Sample mapping from the domain of $f(x)$ to a solution set of 6 supply points. 39

3.16 Forming a pdf approximation: The area under $f(x)$ over an interval is paired with the right-hand endpoint of the interval. . 46

3.17 Level curves in p of the normalized quantization load, for $K = 10, 15, \ldots, 100$. Top: Uniform. Bottom: Triangle. 50

3.18 Level curves in p of the normalized quantization load, for $K = 10, 15, \ldots, 100$. Top: Increasing. Bottom: Decreasing. 51

3.19 Level curves in p of the normalized quantization load, for $K = 10, 15, \ldots, 100$. Top: Unimodal. Bottom: Bimodal. 52

3.20 Bimodal: Level curves in p of the normalized quantization load, for $K = 10, 15, \ldots, 300$. 53

4.1 Rectilinear embedding of the sample problem instance. 61

4.2 A simple circuit with four corners, one of each type. Solid lines indicate a true partition; the dashed lines indicate a false partition. The two $\times$'s mark supply points that do not coincide with demand points. 63

4.3 Possible locations of clause points for the true partition. 66

4.4 Possible locations of clause points for the false partition. 66

4.5 Position of clause point P_1^* for clause $E_1 = (u_1 \vee \overline{u_2} \vee u_3)$, from the example given in Figure 4.1. 67

4.6 Overall cost increases when an illegitimate cluster is chosen. . . 69

4.7 An example of the point numbering scheme. 74

4.8 Lemma 4.2(e): A non-optimal subset of four points that contains both SW corner points has f-value of 22. 76

4.9 Two non-optimal subsets involving clause point P_j^* located in configuration T_1. 79

4.10 Two non-optimal subsets involving clause point P_j^* located in configuration T_2. 80

4.11 Two non-optimal subsets involving clause point P_j^* located in configuration T_3. 80

4.12 Lemma 4.4(g): A third non-optimal subset involving clause point P_j^* located in configuration T_3. 81

4.13 Two non-optimal subsets involving clause point P_j^* located in configuration F_1. 82

4.14 Two non-optimal subsets involving clause point P_j^* located in configuration F_2. 82

4.15 Two non-optimal subsets involving clause point P_j^* located in configuration F_3. 83

5.1 One major iteration of the TB heuristic on a sample problem. . 86

5.2 Scatter plots of $n = 1000$ for the four input distributions. 92

5.3 The normalized ratio vs. p, for the TB and TBr heuristics, for the four input distribution combinations. Each point (with 95% confidence interval) is the mean of 50 problem instances with $n = 100$. 96

5.4 The normalized ratio vs. p, for the TB and TBr heuristics, for the four input distribution combinations. Each point (with 95% confidence interval) is the mean of 50 problem instances with $n = 200$. 97

5.5 (EquallyLikely, EquallyLikely) input: The normalized ratio vs. p, for the TB and TBr heuristics, for problem instances of size $n = 100$ and $n = 200$. Each point (with 95% confidence interval) is the mean of 50 problem instances. 98

5.6 (EquallyLikely, Bimodal) input: The normalized ratio vs. p, for the TB and TBr heuristics, for problem instances of size $n = 100$ and $n = 200$. Each point (with 95% confidence interval) is the mean of 50 problem instances. 99

5.7 (Bimodal, Bimodal) input: The normalized ratio vs. p, for the TB and TBr heuristics, for problem instances of size $n = 100$ and $n = 200$. Each point (with 95% confidence interval) is the mean of 50 problem instances. 100

5.8 (Quadrimodal, Quadrimodal) input: The normalized ratio vs. p, for the TB and TBr heuristics, for problem instances of size $n = 100$ and $n = 200$. Each point (with 95% confidence interval) is the mean of 50 problem instances. 101

5.9 Level curves in p for the (EquallyLikely, EquallyLikely) input combination, $n = 100$. Top: TB. Bottom: TBr. 102

5.10 Level curves in p for the (EquallyLikely, EquallyLikely) input combination, $n = 200$. Top: TB. Bottom: TBr. 103

5.11 Level curves in p for the (EquallyLikely, Bimodal) input combination, $n = 100$. Top: TB. Bottom: TBr. 104

5.12 Level curves in p for the (EquallyLikely, Bimodal) input combination, $n = 200$. Top: TB. Bottom: TBr. 105

5.13 Level curves in p for the (Bimodal, Bimodal) input combination, $n = 100$. Top: TB. Bottom: TBr. 106

5.14 Level curves in p for the (Bimodal, Bimodal) input combination, $n = 200$. Top: TB. Bottom: TBr. 107

5.15 Level curves in p for the (Quadrimodal, Quadrimodal) input combination, $n = 100$. Top: TB. Bottom: TBr. 108

5.16 Level curves in p for the (Quadrimodal, Quadrimodal) input combination, $n = 200$. Top: TB. Bottom: TBr. 109

List of Tables

Table

3.1 Tables used in the Quantize algorithm. 17

3.2 Formulas of pdf and cdf input distributions. 22

3.3 Tables used in the Quantize-Continuous algorithm. 48

4.1 Summary of the reduction from planar 3-SAT to DPM2. 62

4.2 Possible moves in a walk around a circuit. 71

5.1 Probability density function for input distributions. 93

Chapter 1

The p-median problem

1.1 A brief history

The p-median problem is: given n demand points, find p supply points that minimize the sum of the distance from each demand point to its closest supply point, with respect to a particular distance metric. The choice of distance measure impacts the complexity of the problem as well as the approach needed to find a solution. P-median under the Euclidean distance measure has been in existence since at least the 17th century, when Pierre de Fermat[1] posed the 1-median problem with 3 demand points[14]:

> Given three points in the plane, find a fourth point such that the sum of its distances to the three given points is a minimum.

At the start of the twentieth century, in one of the founding texts in location theory, Alfred Weber considered a version of the Euclidean 1-median problem to determine industrial location while minimizing transport cost [27]. Writing

[1]The origin of the problem is a matter of debate. See [11] for a historical review of the 1-median (Weber) problem.

in 1977, Ostresh noted that the problem "has application to the siting of steel mills and schools, houses of ill repute and hospitals" [19]. More recently, p-median has arisen in areas such as statistical cluster analysis, spatial data mining, and data compression.

In this work, we consider the p-median problem under a new distance metric, the directional rectilinear distance. In later chapters we will consider both the single and multiple dimensional cases of the directional p-median problem, giving optimal solutions for the former and heuristic solutions for the latter. But first, we present in this chapter an overview of the p-median problem under traditional distance measures and the definition of the directional rectilinear metric.

1.2 The p-median problem: continuous and discrete space

In d-dimensional space, $d \geq 2$, the **continuous** p-median problem allows supply points to be located anywhere in d-space, not merely from among the given demand points. The **discrete** problem provides a list of **candidate points** from which supply points may be chosen. Let $d((x_i, y_i), (z_j, t_j))$ be the distance from point (x_i, y_i) to point (z_j, t_j) according to some distance metric. The decision version of the continuous p-median problem in the plane may be formally stated as:

Problem 1.1 (Continuous-PM2) Given a set $X = \{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\}$ of demand points in the plane, an integer p, and a bound B, does there exist a set

$$S = \{(z_1, t_1), (z_2, t_2), \ldots, (z_p, t_p)\}$$

of p supply points such that

$$\sum_{i=1}^{n} \min_{1 \le j \le p} \{d((x_i, y_i), (z_j, t_j))\} \le B \, ?$$

Continuous-PM2 is NP-complete under either the Euclidean (d_e) or the rectilinear (d_r) distance measure [17], where d_e and d_r are defined as:

$$\begin{aligned} d_e((x_i, y_i), (z_j, t_j)) &= \sqrt{(x_i - z_j)^2 + (y_i - t_j)^2} \\ d_r((x_i, y_i), (z_j, t_j)) &= |x_i - z_j| + |y_i - t_j| \end{aligned}$$

Under the rectilinear distance measure, it is well known that only demand points and **intersection points** need be considered as candidates for supply points. Intersection points are found by crossing the set $\{x_1, x_2, \ldots, x_n\}$ with the set $\{y_1, y_2, \ldots, y_n\}$, and subtracting the demand points, yielding at most $n^2 - n$ new points. Thus the continuous p-median problem under the rectilinear distance measure reduces to a discrete p-median problem (see [5] for a complete treatment of discrete location problems).

The discrete p-median problem in the plane can be formulated as the following integer program [22]:

Problem 1.2 (Discrete-PM2) Minimize

$$\sum_{i \in X} \sum_{j \in C} d_{ij} r_{ij}$$

subject to

$$\begin{aligned}
\sum_{j \in C} r_{ij} &= 1 \qquad \forall i \in X, \\
r_{ij} &\leq s_j \qquad \forall i \in X,\ j \in C, \\
\sum_{j \in C} s_j &= p, \\
r_{ij},\ s_j &\in \{0, 1\} \qquad \forall i \in X,\ j \in C,
\end{aligned}$$

where

$$\begin{aligned}
i \in X &= \text{the set of demand points,} \\
j \in C &= \text{the set of candidate points,} \\
d_{ij} &= \text{distance from point } i \text{ to point } j, \\
p &= \text{number of supply points to be chosen,} \\
r_{ij} &= \begin{cases} 1 & \text{if point } i \text{ is assigned to candidate } j, \\ 0 & \text{otherwise,} \end{cases} \\
s_j &= \begin{cases} 1 & \text{if candidate } j \text{ is chosen,} \\ 0 & \text{otherwise.} \end{cases}
\end{aligned}$$

The distances between demand points and candidate points are organized into a distance matrix $[d_{ij}]$. In Section 5.1 we discuss properties of the distance matrix that affect the difficulty of finding a good quality solution [25].

Figure 1.1 shows the 2-dimensional p-median problem under three different combinations of the solution space (continuous or discrete) and distance measure (Euclidean or rectilinear).

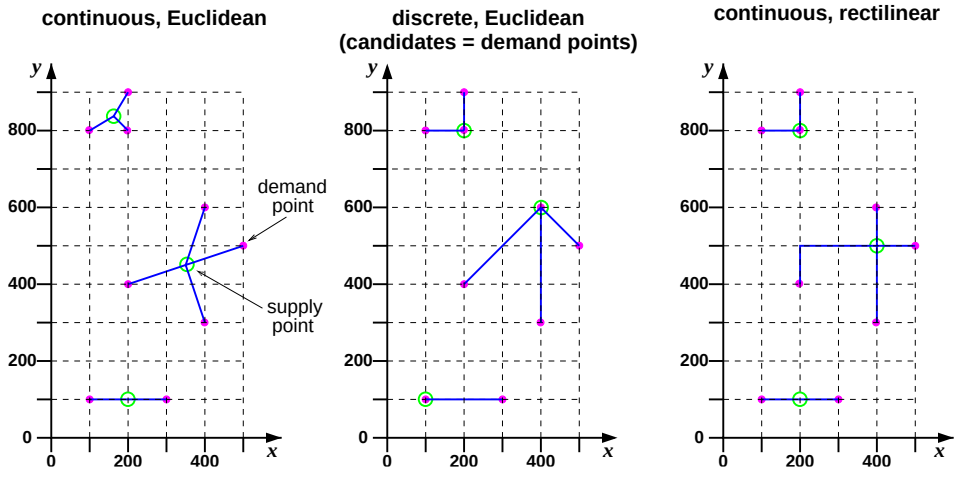


Figure 1.1: Sample p-median problem instances in the plane, with $n = 9$ demand points and $p = 3$ supply points.

Summary of complexity results

In one dimension, the rectilinear and Euclidean distance measures are the same. Hassin & Tamir demonstrate that the one-dimensional p-median problem is solved in time $O(pn)$ [13].

In two or more dimensions, p-median is an NP-complete problem under either the rectilinear or the Euclidean distance measure [17]. Chapter 5 discusses a few of the many heuristics developed for this problem.

1.3 A new notion of distance: The directional distance metric

We now define the **directional** rectilinear distance measure. In general, a c-directional, d-dimensional rectilinear metric (with $c \leq d$) defines distance from point $(p_1, \ldots, p_d)$ to $(q_1, \ldots, q_d)$ to be ∞ if $p_i > q_i$ for some $i \in \{1, \ldots, c\}$ and $\sum_{1 \leq i \leq d} |q_i - p_i|$ otherwise. Thus, in a directional p-median problem, a supply point must achieve or exceed the values of the first c coordinates of all its demand points.

In Chapter 4 we show that the rectilinear c-directional, d-dimensional p-median problem is NP-complete when $c = d = 2$ (problem DPM2), which implies NP-completeness for all c, d satisfying $2 \leq c \leq d$. In the plane, the 2-directional rectilinear distance is:

$$d_{dr}((x_i, y_i), (x_j, y_j)) = \begin{cases} x_j - x_i + y_j - y_i & \text{if } x_j \geq x_i \text{ and } y_j \geq y_i, \\ \infty & \text{otherwise} \end{cases}$$

We define **directional intersection points** to be the subset of intersection points that lie above (at least) one demand point as well as to the right of (at least) one demand point. Specifically, the point (x_j, y_j) is a directional intersection point if:

(1) $(x_j, y_j) \notin X$, that is, (x_j, y_j) is not itself a demand point, and

(2) there exist points $(x_i, y_i) \in X$ and $(x_k, y_k) \in X$ for which $x_j = x_i$ and $y_j = y_k$ and $x_j > x_k$ and $y_j > y_i$.

Figure 1.2 shows the worst case scenario, an instance of DPM2 which has the most directional intersection points possible, $(n^2 - n)/2$.

1.4 Organization of thesis

Chapter 2 describes two main applications of directional p-median. Chapter 3 considers the one-dimensional problem, directional p-median on the real line, which is solvable in polynomial time. The next two chapters consider the multidimensional problem: Chapter 4 proves it NP-complete and Chapter 5 presents a fast heuristic algorithm for solving it. We conclude in Chapter 6 and outline some areas for future work.

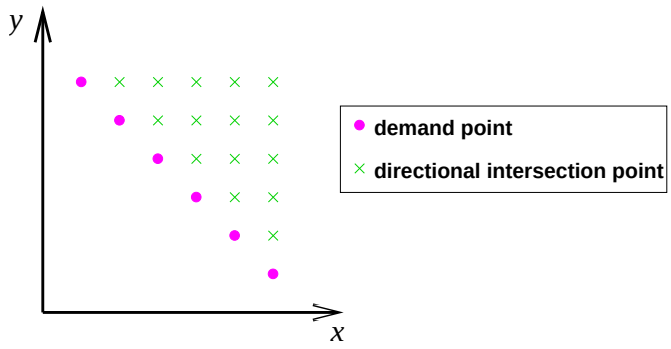


Figure 1.2: Demand points arranged in a downward slope yield the maximum number of directional intersection points.

Chapter 2

Applications of the directional p-median problem

We motivate the directional p-median problem by presenting two sample applications. As compared to many facility location problems found in the literature, the applications described here will have a much larger number of candidate points, *e.g.* thousands or millions instead of hundreds.

2.1 Traffic quantization

The directional p-median problem arises in any network environment in which a number of lower rate streams are multiplexed for transport over a higher rate channel. In the emerging optical Wavelength Division Multiplexing (WDM) networks, where the data rate of a single channel is 2.5 to 10 *Gbps*, each channel is capable of carrying hundreds, even thousands, of independent data streams. Managing and controlling such a large number of traffic streams at Gigabit per second data rates is an extremely complicated task, while the

complexity of certain functions (*e.g.*, Quality of Service scheduling) increases faster than linearly with the number of data streams.

The multidimensional directional p-median problem corresponds to demand requests characterized by two or more parameters. For example, a customer of computing services may request service for various Quality of Service (QoS) "resources", such as bandwidth, burst size, or delay bounds. Rather than meeting each customer's demand with a unique level of service, the service provider may prefer to group (quantize) similar requests into a single service level, in such a way that any given customer receives *at least* the amount requested of a given resource. In the plane, a given demand point will be mapped to a level of service (a supply point) that is located above and to the right of it, corresponding to a c-directional, d-dimensional distance metric with $c = d = 2$. The directional *rectilinear* metric is appropriate whenever cost of service is a linear combination of costs along each dimension. The operations of traffic engineering, packet scheduling and QoS support, network management, traffic policing, and billing are greatly simplified in a such a network.

Performance analysis is also more tractable in a system offering a limited number of service levels, since systems with continuous rates give rise to analytical models with infinite dimensions (note also that these models are usually approximated by finite-dimensional ones). On the other hand, limiting the number of supported rates does have a disadvantage in that it may use more resources (*e.g.*, network bandwidth) than a continuous-rate system to accommodate a given set of customer requests. Rather than receiving the exact rate needed, a request may have to subscribe to the next higher rate offered. As a result, quantization will have an adverse effect on system performance resulting in either a higher blocking probability (*i.e.*, a higher probability of

denying a request compared to a continuous-rate system employing the same amount of resources) or a lower utilization (since more resources may be needed to carry the same set of requests as a continuous-rate system).

Lea and Alyatama consider the problem of quantizing packet traffic in the context of ATM networks, demonstrating that ATM networks offering a handful of quantized levels suffer little performance degradation compared to continuous rate networks [15]. Our conclusions are similar although our approach is different and our results are stronger. Specifically, [15] takes a queueing theoretic approach, considering a single link with Poisson arrivals, and uses a heuristic technique (simulated annealing) to obtain a sub-optimal vector of service levels. In Chapter 3 we use a dynamic programming approach which allows us to compute the *optimal* service levels in a very efficient manner.

2.2 Multiprocessor scheduling of periodic tasks

Like data streams vying for data channels, periodic tasks in a system of multiple identical processors must contend for the limited resources of processor time. Tasks must be scheduled ("multiplexed") such that their deadlines are respected. In a multiprocessor system that provides only a (small) set of quantized service levels, many functions, such as billing and the scheduling, management, and handling of dynamic task requests, will be significantly simplified as compared to a system offering a continuous spectrum of rates.

In particular, we show in Section 3.6 that the preemptive scheduling of a set of n periodic tasks on m identical processors is much easier when the task set has been quantized. In this slotted time model, a periodic task (demand point) is characterized by a rational density x_i, $0 < x_i < 1$, such that if x_i is

written as a fraction in lowest terms, $x_i = \frac{C_i}{D_i}$, then C_i is the computation time (number of subtasks) that must be processed within the period D_i.

The system is subject to two constraints: the **Processor Constraint** requires that at any instant in time, a processor may work on at most one subtask, and the **Task Constraint** requires that at any instant in time, a task may have a subtask being processed by at most one processor. This model is nearly identical to that considered in [4] and [7], which does not require the ratio $\frac{C_i}{D_i}$ to be in lowest terms.

A feasible schedule for this problem exists if and only if $\sum_{i=1}^{n} x_i \leq m$ [4], and the fastest scheduling algorithm runs in time $O(m \log n)$ at each slot [3]. The priority of competing subtasks is determined in part by a subtask's slot deadline, the latest slot in which it may be scheduled. Since the algorithm at each slot selects the m tasks with the most imminent slot deadlines, the running time per slot can be no lower than $O(m \log n)$. Therefore, this algorithm may not be appropriate for applications with a very large number of tasks (n). For instance, consider a web server for a popular web site which uses multiple processors to serve client requests. Such a web site may receive millions of requests per minute; therefore, it is essential to have a scheduling algorithm with a running time independent of the number of requests. Also, consider the recent announcement by IBM regarding the creation of server farms that will provide processing power to applications on demand. This view of processing power as a service that is provided by some form of public utility may be appealing to both individuals and companies of all sizes (which may wish to reduce costs by outsourcing their computation needs much like they "outsource" their power or water needs). Such a public utility will face very large task sets that are also highly dynamic in nature. Thus, it will have

to rely on fast scheduling algorithms in order to provide service in an effective and efficient manner.

As mentioned in Section 2.1, quantization will have an adverse effect on performance, either a higher blocking probability compared to a continuous-rate system, or a lower utilization, since a larger number of processors may be needed to carry the same set of tasks. However, in applications where n is very large, we believe that the performance penalty is more than offset by the improvements in speed and complexity resulting from quantization (as shown in Section 3.6).

Chapter 3

The directional p-median problem on the real line

Throughout this chapter, we occasionally refer to the second sample application from Section 2.2, the scheduling of periodic tasks on a multiprocessor. Without loss of generality, we assume that each demand point x_i is a rational number on the interval $(0, 1)$, with the interpretation that x_i is the requested rate of processor share which cannot exceed 1 (representing 100% utilization of a processor). In Section 3.1, we begin with the formal definition of the directional p-median problem on the real line (DPM1), and then present an optimal dynamic programming solution. In Section 3.3, we present the results of a simulation study that examines the tradeoff penalty due to quantization, in terms of excess resources needed to meet the demands of a set of quantized requests. We show how to formulate DPM1 as a special case of the constrained shortest path problem in Section 3.4. In Section 3.5, we introduce the stochastic directional p-median problem (SDPM1), for which the input is a probability density function describing the population of demand requests.

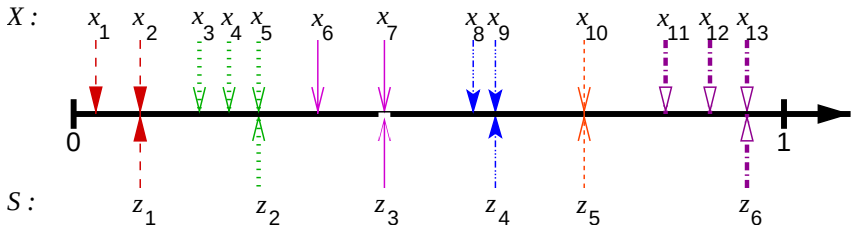


Figure 3.1: Sample mapping of task densities to service levels.

We present an optimal solution for a certain class of probability density functions, as well as an approximate solution. Finally, in Section 3.6, we simplify the multiprocessor scheduling algorithm from [2] by restricting the number of service rates (supply points) offered.

3.1 Statement of the problem: DPM1

Let X be a set of n demand points on the real line $\{x_1, \ldots, x_n\}$, such that $x_1 \leq x_2 \leq \cdots \leq x_n$, and let the density of X be $\rho_X = \sum_{i=1}^{n} x_i$. A set of supply points $S = \{z_1, \ldots, z_p\}$, $z_1 < z_2 < \cdots < z_p$, $1 \leq p \leq n$, is a **feasible solution** for X if and only if $x_i \leq z_p$, $i = 1 \ldots n$. For notational convenience, we assume $z_0 = 0$. Associated with a feasible solution is an **implied mapping** from $X \rightarrow S$, where $x_i \rightarrow z_j$ if and only if $z_{j-1} < x_i \leq z_j$.

Figure 3.1 shows a sample mapping from a set of 13 demand points onto a solution set of 6 supply points. Let X_j be the set of demands mapped to supply point z_j, and let $|X_j| = n_j$.

Problem 3.1 (DPM1) Given a set X of n demand points, $x_1 \leq x_2 \leq \cdots \leq x_n$, find a feasible set S of p supply points, $z_1 < z_2 < \cdots < z_p$, $1 \leq p \leq n$, which

minimizes the following objective function:

$$
\begin{aligned}
g_D(z_1, \ldots, z_p) &= \sum_{j=1}^{p} \sum_{x_i \in X_j} (z_j - x_i) && (3.1)\\
&= \sum_{j=1}^{p} (n_j z_j) - \rho_X && (3.2)\\
&= q_D(z_1, \ldots, z_p) - \rho_X && (3.3)
\end{aligned}
$$

The objective function g_D represents the performance penalty due to quantization. That is, g_D is the amount of excess resources (in our example, excess processor load) needed for the system to service the set of quantized demands, which results after the original demand set has been mapped onto a solution set of p supply points.

The second term of Equation (3.3), ρ_X, the **requested load**, is the amount of load requested by the original set of demand points, while the first term, $q_D(z_1, \ldots, z_p)$, is the **quantization load**, the load assigned to the set of quantized points.[1] The minimum (optimal) value of g_D is called g_D^*, and a feasible set S at which g_D^* is obtained is called an **optimal** solution set for X. Minimizing g_D also minimizes a quantity called the Normalized Quantization Load for deterministic input, NQL_D:

$$
\begin{aligned}
NQL_D &= \frac{q_D(z_1 \ldots z_p)}{\rho_X} && (3.4)\\
&= \frac{\sum_{j=1}^{p} n_j z_j}{\rho_X} && (3.5)\\
&\geq 1 && (3.6)
\end{aligned}
$$

Clearly, the closer NQL_D is to 1, the fewer resources are wasted.

[1]The subscript "D" in $g_D(z_1, \ldots, z_p)$ and $q_D(z_1, \ldots, z_p)$ stands for deterministic input. We will derive similar expressions for stochastic input in Section 3.5.

For any feasible solution set for which $x_n < z_p$, the objective function g_D can be reduced by setting $z_p = x_n$. Therefore in an optimal solution set, the maximum supply point z_p must equal the maximum demand point x_n. Furthermore, we can state the following lemma:

Lemma 3.1 Let X be a set of n demand points such that $x_1 \leq x_2 \leq \cdots \leq x_n$. There exists an optimal solution set $S = \{t_1, \ldots, t_p\}$, $t_1 < t_2 < \cdots < t_p$, of X, for which $t_j \in X$, for each $j = 1, \ldots, p$.

Proof. Suppose there exists an optimal solution set of X called $S_1 = \{z_1 \ldots z_p\}$, $z_1 < z_2 < \cdots < z_p$, for which there exists some $z_a \in S_1$ but $z_a \notin X$. There can be no x_i that is mapped to z_a. If there were, then z_a could be moved down to $z_a - \Delta$, for some $\Delta > 0$, and the objective function could be lowered, contradicting the optimality of S_1. Therefore we can create S from S_1 by setting $t_j = z_j$ for $j \neq a$, and $t_a = x_n$. ■

3.2 A dynamic programming algorithm for DPM1: Quantize

We now present the algorithm Quantize which uses a dynamic programming approach to obtain an optimal set of supply points for problem DPM1. This approach is based on the observation that, due to Lemma 3.1, an optimal set of supply points is a subset of the set X of demand points. Quantize computes this optimal subset in an efficient manner.

Quantize builds four tables of values described in Table 3.1, and in doing so finds the optimal value of the objective function g_D^* and an optimal solution set S. The $n \times n$ tables `Diff` and `Cumul` hold differences and cumulative sums of differences, respectively, values that will be used to fill in the entries

name	size	defined	description of entry i, j
`Diff`	$n \times n$	$i \leq j$	the difference `x[`j`]`$-$`x[`i`]`
`Cumul`	$n \times n$	$j \leq i$	sum of entries in column i of `Diff`, from row j to row i
`Opt`	$n \times p$	$j \leq i$	minimum value of g_D for DPM1 with $X = \{x_1, \ldots, x_i\}$ and $p = j$
`Prev`	$n \times p$	$j \leq i$ and $j \neq 1$	index into `x` array of z_{j-1}, if $z_j =$`x[`i`]`.

Table 3.1: Tables used in the Quantize algorithm.

of the $n \times p$ table `Opt`. Entries in `Diff` and `Cumul` are calculated according to the following formulas (the array `x` holds the elements of the task set X):

$$\begin{aligned} \texttt{Diff}[i][j] &= \texttt{x}[j] - \texttt{x}[i] \ , \ i \leq j \\ \texttt{Cumul}[i][j] &= \sum_{k=j}^{i} \texttt{Diff}[k][i] \\ &= \sum_{k=j}^{i} (\texttt{x}[i] - \texttt{x}[k]) \ , \ j \leq i \end{aligned}$$

Filling in a single entry of `Opt` corresponds to solving one instance of DPM1; entry (i,j) holds the minimum value of the objective function g_D for an instance of DPM1 in which $X = \{x_1, \ldots, x_i\}$ and $p = j$. Each entry of `Opt` is calculated recursively, using entries representing smaller problem instances, that is, an instance having a smaller value of n or a smaller value of p or both. Specifically:

$$\texttt{Opt}[i][j] = \begin{cases} 0 & \text{if } i = j \\ \texttt{Cumul}[i][j] & \text{if j} = 1 \\ \min_{k=j-1}^{i-1} \{\texttt{Opt}[k][j-1] + \texttt{Cumul}[i][k+1]\} & j < i \text{ and } j \neq 1 \end{cases} \tag{3.7}$$

Lastly, the $n \times p$ table `Prev` holds the information needed to construct the optimal solution set S. By Lemma 3.1, each supply point $z_j \in L$ must take on the value of some $x_i \in X$. `Prev`$[i][j]$ holds the index into the `x` array of z_{j-1}, assuming that $z_j = x_i$; that is, if $z_j = x_i$, then $z_{j-1} =$ `x`[`Prev`$[i][j]$]. `Prev`$[i][j]$ also equals the value of k at which the minimum was attained in the third line of Equation (3.7) (or $i - 1$ if $i = j$). Note that for $j = 1$, `Prev`$[i][j]$ is undefined, since there is no z_0. Thus when Quantize finishes building `Prev`, the optimal solution set S can be constructed using only a few lines of code. The pseudocode description of Quantize can be found in Appendix A.

Correctness proof

Theorem 3.4 below proves the correctness of Quantize by demonstrating that the value calculated for `Opt`$[n][p]$ is equal to the optimal value of the objective function g_D for an instance of DPM1 in which a set of n tasks are optimally quantized (mapped) onto p supply points. We first prove two lemmas to aid in the proof of Theorem 3.4.

Lemma 3.2 `Opt`$[n][p] = g_D^*(z_1, \ldots, z_p)$ whenever $p = n$.

Proof. Since there are as many supply points as there are demand points, each demand point receives exactly the amount of resources it requests; that is, $z_i = x_i$ for $i = 1, \ldots, n$. Thus $g_D^*(z_1, \ldots, z_p) = \sum_{i=1}^{n} (z_i - x_i) = 0$, which agrees with the first case of Equation (3.7). ■

Lemma 3.3 `Opt`$[n][p] = g_D^*(z_1, \ldots, z_p)$ whenever $p = 1$.

Proof. Since there is only one supply point, then $z_1 = x_n$, and thus $g_D^*(z_1) = \sum_{i=1}^{n}(x_n - x_i) = \texttt{Cumul}[n][1]$, which agrees with the second case of Equation (3.7). ■

Theorem 3.4 $\texttt{Opt}[n][p] = g_D^*(z_1, \ldots, z_p)$, for $n \geq 1$ and $1 \leq p \leq n$.

Proof. By induction. For the base case, let $n = 1$ and $p = 1$. By Lemma 3.2, $\texttt{Opt}[1][1] = g_D^*(z_1)$.

Assume $\texttt{Opt}[i][j] = g_D^*(z_1, \ldots, z_p)$, for all possible (i,j)-pairs where $i = 1, \ldots, n$ and $j = 1, \ldots, p$, for some $n \geq 1$ and $1 \leq p < n$. We now prove that $\texttt{Opt}[n][p+1] = g_D^*(z_1, \ldots, z_{p+1})$. Note that this is sufficient for the induction step; by Lemma 3.3 we can always fill in the first element of each row of the Opt table. Then we need only fill in each row from left to right, beginning with row 1 and proceeding to row 2, *etc.* Thus for the induction step it is sufficient to show that we can accurately calculate the next entry (one column to the right) in the current row, namely $\texttt{Opt}[n][p+1]$.

Case (a): $p + 1 = n$. By Lemma 3.2, $\texttt{Opt}[n][p+1] = g_D^*(z_1, \ldots, z_p)$.

Case (b): $p + 1 < n$. The largest supply point must equal the largest demand point: $z_{p+1} = x_n$. We next examine all possible values for z_p and choose the one that yields the lowest value of the objective function $g_D(z_1, \ldots, z_{p+1})$. According to Lemma 3.1, we need only consider the demand points as possible values for z_p; in particular, z_p may only equal one of the following demands: $\{x_p, x_{p+1}, \ldots, x_{n-1}\}$. Suppose $z_p = x_k$, for some $k \in \{p, p+1, \ldots, n-1\}$. Then the demands $\{x_{k+1}, \ldots, x_n\}$ will be mapped to $z_{p+1} = x_n$, and the contribution to the objective

function $g_D(z_1, \ldots, z_{p+1})$ from the z_{p+1}-mapping alone will be:

$$\begin{aligned}\sum_{i=k+1}^{n} (z_{p+1} - x_i) &= \sum_{i=k+1}^{n} (x_n - x_i) \\ &= \sum_{i=k+1}^{n} \texttt{Diff}[i][n] \\ &= \texttt{Cumul}[n][k+1]\end{aligned}$$

This quantity, $\texttt{Cumul}[n][k+1]$, is exactly the second term inside the min function of the third case of Equation (3.7). Next, we calculate the contribution to the objective function $g_D(z_1, \ldots, z_{p+1})$ from the demand mappings to the remaining supply points z_1 to z_p; this contribution depends on the placement of z_1 to z_{p-1} (recall that we have fixed z_p at x_k). By the inductive hypothesis, we have already calculated the optimal placement of z_1 to z_{p-1} whenever $z_p = x_k$; in particular, $\texttt{Opt}[k][p]$ is the minimum value of the objective function for a problem instance in which $X = \{x_1, \ldots, x_k\}$ is mapped onto p supply points, and the `Prev` table holds the positions of the supply points that yield this minimum value. Thus the min function of the third case of Equation (3.7) does the following: for each possible position x_k for z_p, $x_k \in \{x_p, \ldots, x_{n-1}\}$, it calculates the objective function $g_D(z_1, \ldots, z_{p+1}) = \texttt{Opt}[k][p] + \texttt{Cumul}[n][k+1]$, and then chooses the position that yields the minimum. The chosen value of k is stored in the `Prev` table. ■

Analysis of Quantize

The operation of Quantize can be divided into four sequential tasks. First, the algorithm builds the `Diff` table in time $O(n^2)$. Second, it uses the `Diff` table to fill in the entries of the `Cumul` table, also in time $O(n^2)$. Third, the algorithm builds the `Opt` table: for an entry calculated using the third line of Equation (3.7), the min operation inspects at most n sums; hence the line with the min operation requires time $O(n)$. There are at most np entries in `Opt`, and thus Quantize builds the `Opt` table in time $O(n^2p)$. The fourth and final task of Quantize consists of constructing the optimal solution set S from the information held in the `Prev` table, which is accomplished in time $O(p)$. Therefore, the overall running time of Quantize is $O(n^2 + n^2 + n^2p + p)$ or $O(n^2p)$.

3.3 Performance penalty due to quantization for Quantize

Simulation set-up and input parameters

To determine the penalty in terms of excess resources needed as a result of quantization, a simulation study was designed using a variety of different types of demand sets X. In particular, six different input distributions were used to generate demand sets X in the simulations: uniform, triangle, increasing, decreasing, unimodal, and bimodal. Figures 3.2-3.4 show the graph of each input distribution's probability density function. The mathematical expressions of each probability density function (pdf) and cumulative distribution function (cdf) are given in Table 3.2. From each input distribution, one hundred demand sets with $n = 100$ were generated, and another one hundred demand sets with $n = 1000$ were generated. Each demand set was gener-

Distribution	$f(x)$	$F(x)$	domain
Uniform	1	x	$0 < x < 1$
Triangle	$4x$	$2x^2$	$0 < x < .5$
	$-4x + 4$	$-2x^2 + 4x - 1$	$.5 \leq x < 1$
Increasing	$2x$	x^2	$0 < x < 1$
Decreasing	$-2x + 2$	$-x^2 + 2x$	$0 < x < 1$
Unimodal	$4/9$	$4x/9$	$0 < x < .25$
	6	$6x - 25/18$	$.25 \leq x < .35$
	$4/9$	$4x/9 + 5/9$	$.35 \leq x < 1$
Bimodal	$1/4$	$x/4$	$0 < x < .25$
	4	$4x - 15/16$	$.25 \leq x < .35$
	$1/4$	$x/4 + 3/8$	$.35 \leq x < .65$
	4	$4x - 33/16$	$.65 \leq x < .75$
	$1/4$	$x/4 + 3/4$	$.75 \leq x < 1$

Table 3.2: Formulas of pdf and cdf input distributions.

ated starting from a unique seed for a Lehmer random number generator with modulus $2^{31} - 1$ and multiplier 48271.

Each demand set then served as input to the algorithm Quantize. We used the normalized quantization load NQL_D (defined in Equation (3.4)) as the measure of the performance penalty due to quantization. For each demand set, NQL_D was calculated for a variety of values of p, the number of supply points in the optimal solution set.

Simulation results

Figures 3.5-3.7 contain a total of six graphs, one for each input distribution. Each graph shows NQL_D along the y-axis corresponding to values of p ranging from $p = 2, 3, \ldots, 100$ along the x-axis, for demand sets of size $n = 100$ and $n = 1000$. Each point was generated by averaging NQL_D across one hundred demand sets. The $n = 1000$ curve lies slightly above the $n = 100$ curve, yet the general shape of the curves remains the same regardless of X and

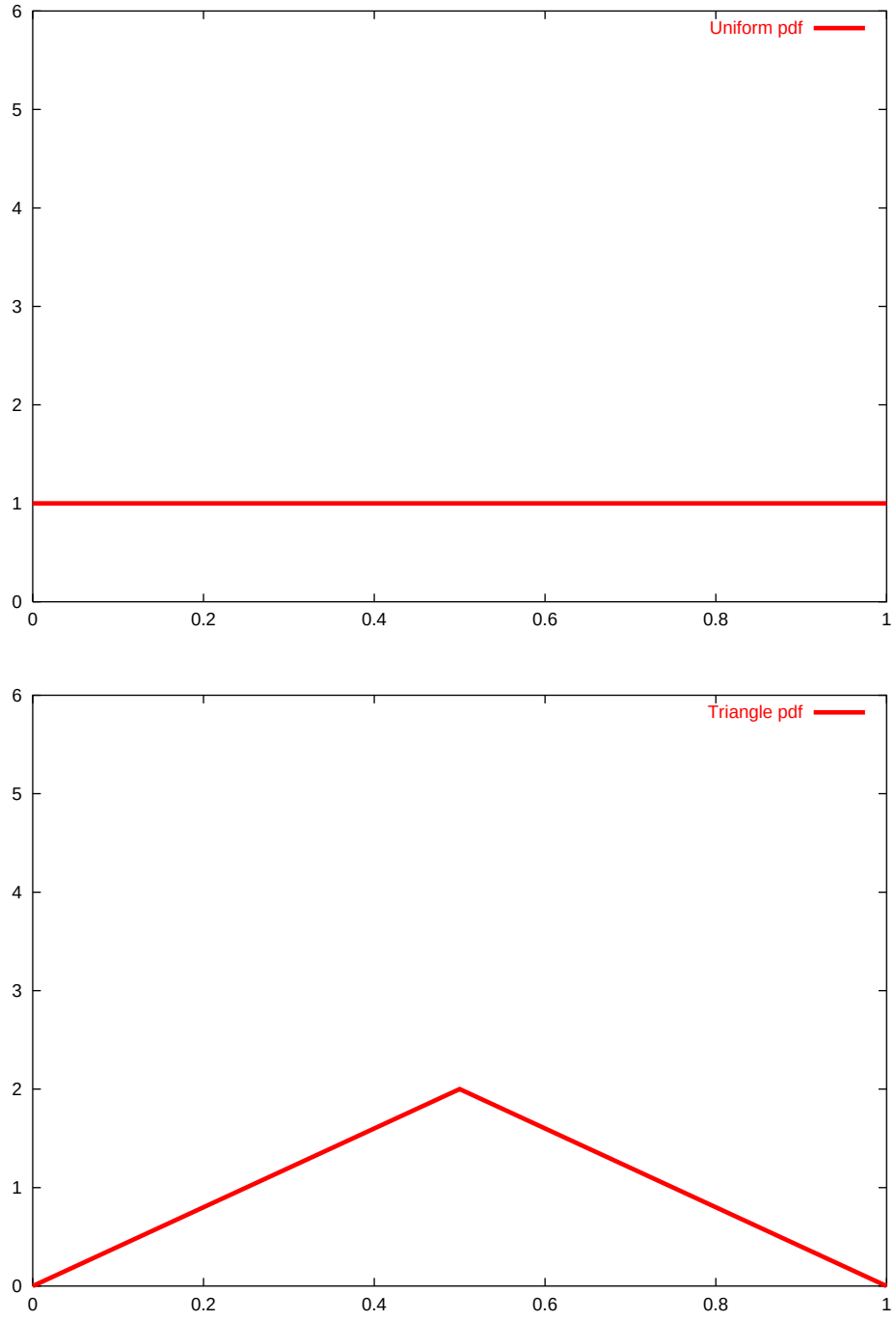


Figure 3.2: Probability density functions for the uniform and triangle input distributions.

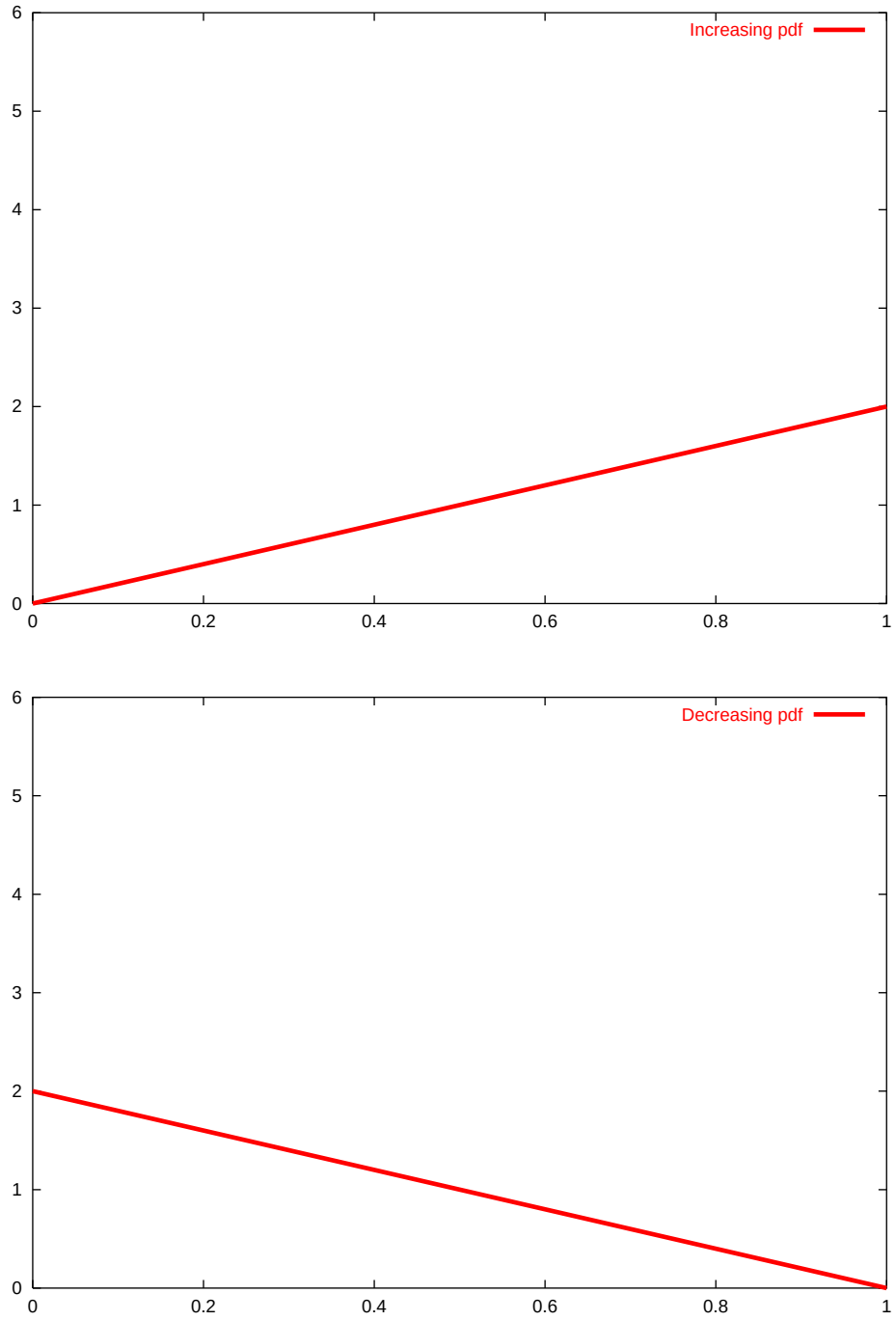


Figure 3.3: Probability density functions for the increasing and decreasing input distributions.

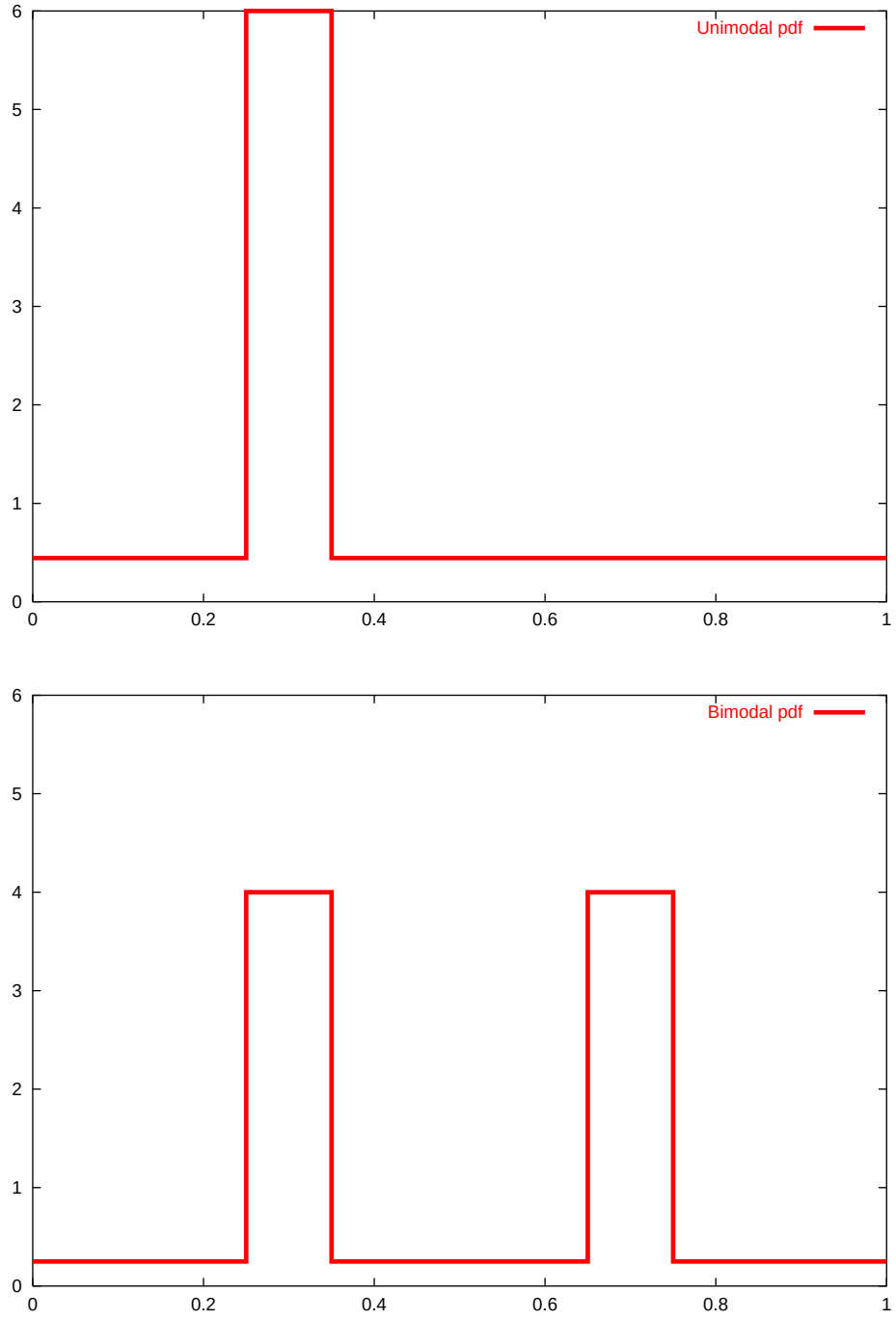


Figure 3.4: Probability density functions for the unimodal and bimodal input distributions.

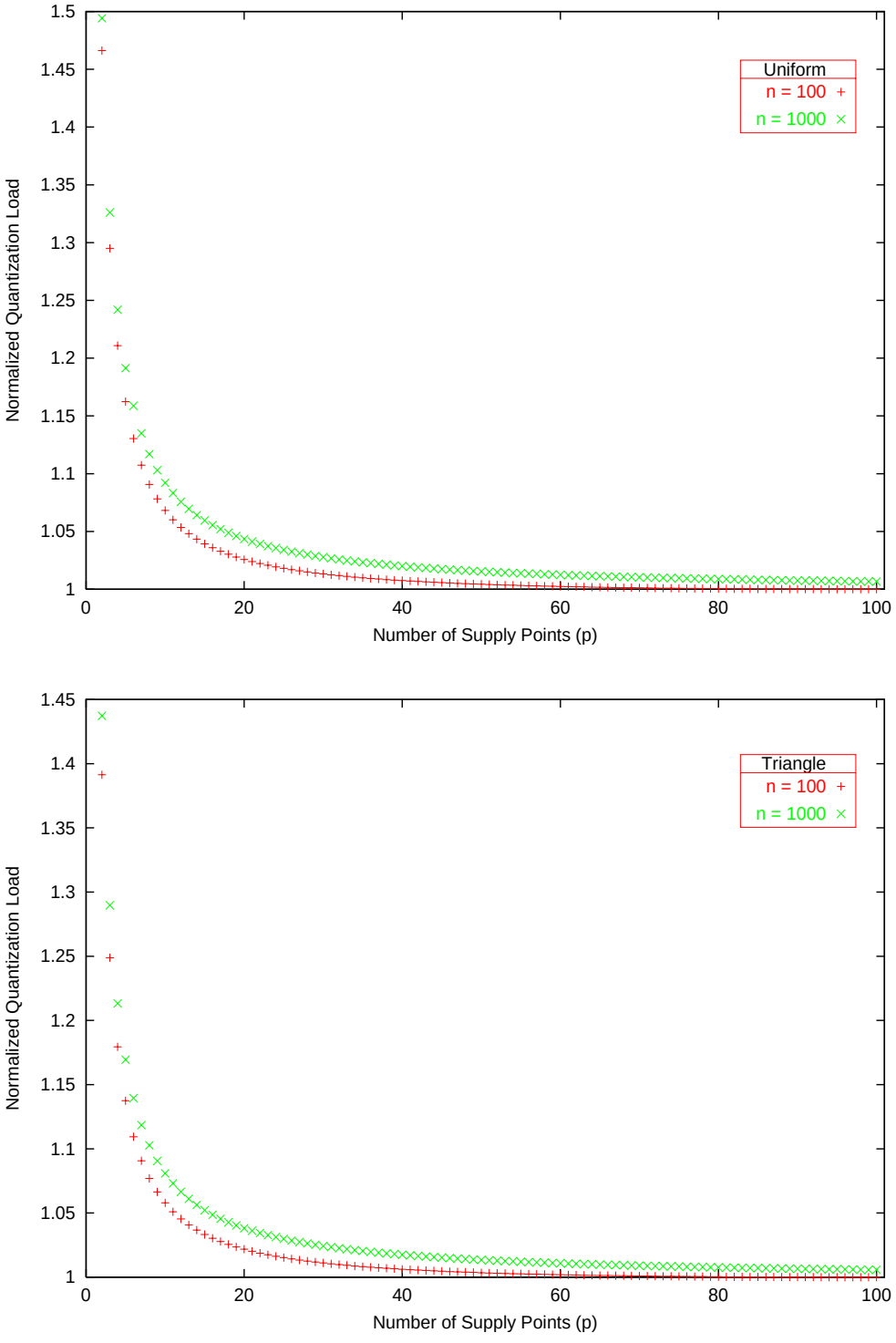


Figure 3.5: Level curves in n of the normalized quantization load vs. number of supply points (p). Top: Uniform. Bottom: Triangle.

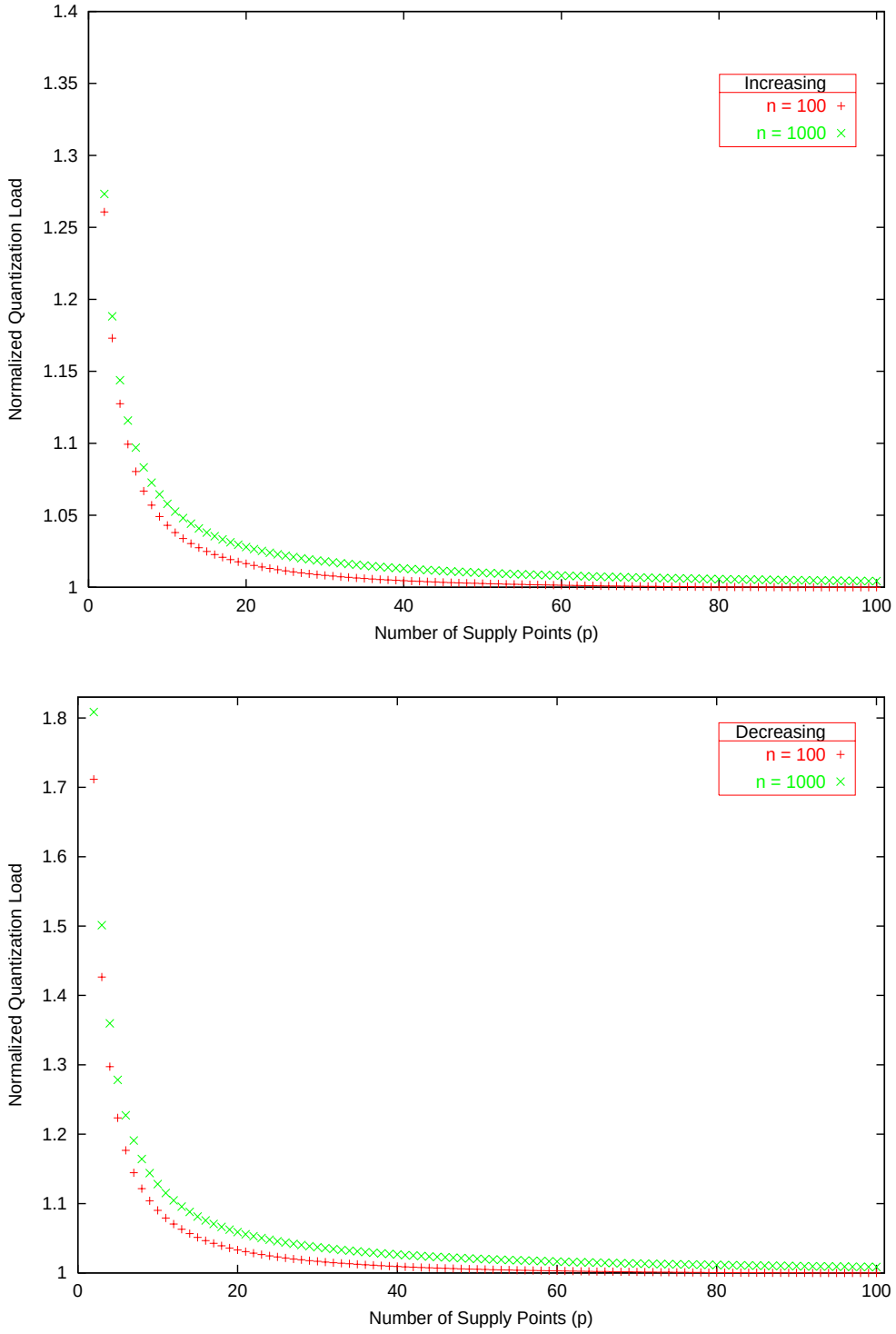


Figure 3.6: Level curves in n of the normalized quantization load vs. number of supply points (p). Top: Increasing. Bottom: Decreasing.

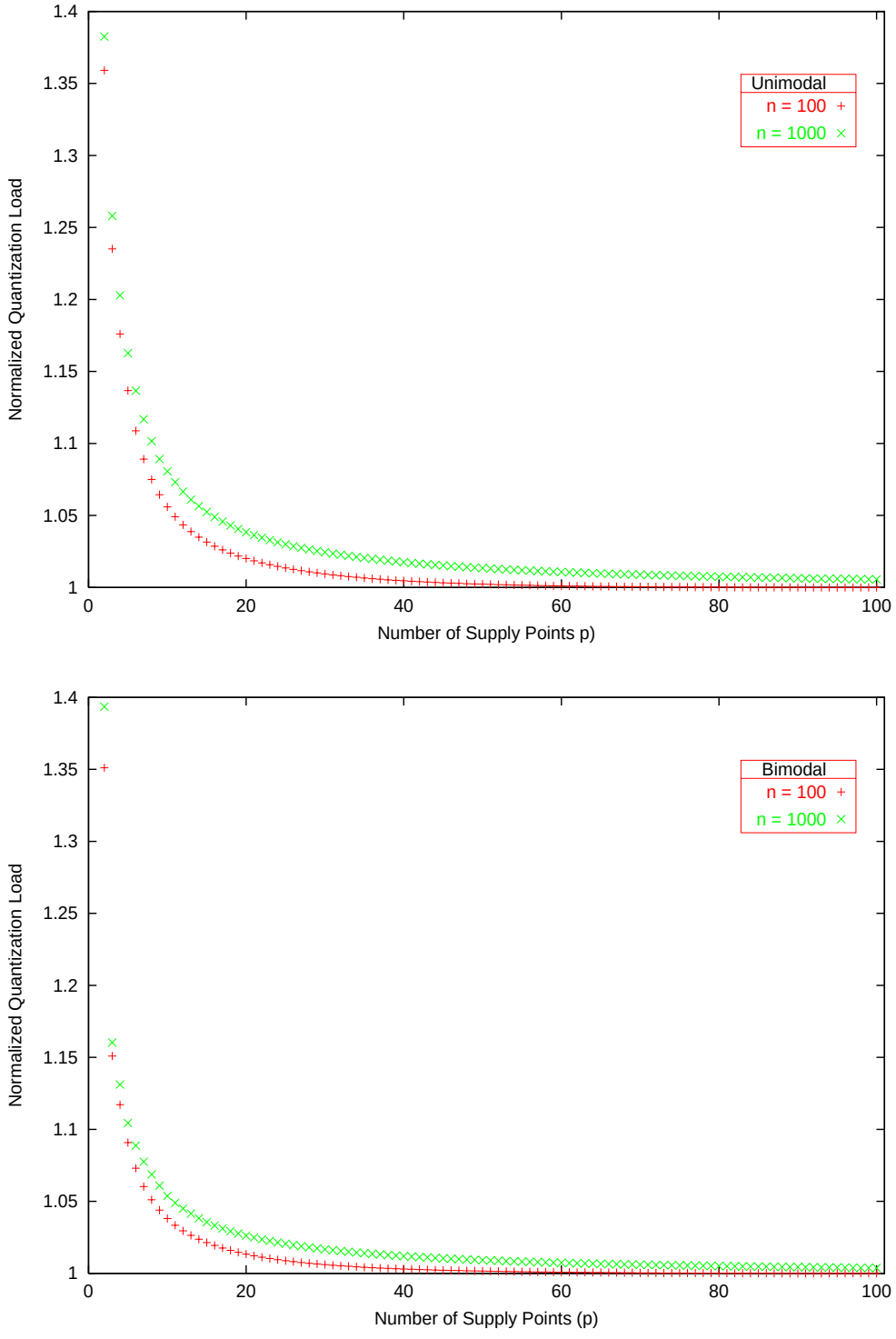


Figure 3.7: Level curves in n of the normalized quantization load vs. number of supply points (p). Top: Unimodal. Bottom: Bimodal.

input distribution: NQL_D drops immediately as p increases. In each graph, NQL_D has dropped below 1.05 at an p-value ≤ 20, for both the $n = 100$ curve and the $n = 1000$ curve. In the multiprocessor example, this result means that by using only 20 (or fewer) rate levels (supply points), we can adequately service demand sets of 100 or even 1000 rate requests, dedicating no more than 5% processor resources beyond the amount requested. Another interpretation is that, for a fixed amount of processor resources, we can accept rate requests up to approximately 95% capacity.

Figure 3.8 contains two graphs using the uniform input distribution. The top graph shows NQL_D along the y-axis corresponding to each of the one hundred individual demand sets X for $n = 100$. The bottom graph shows the same, for one hundred demand sets with $n = 1000$. Level curves for $p = 2, 4, 6, 8, 10, 15$ and 20 are shown. These graphs present the information contained within the uniform-input graph of Figure 3.5 in a different way; the single point at, say, $p = 10$ in the $n = 100$ (respectively, $n = 1000$) uniform-input graph of Figure 3.5 was created from averaging the NQL_D values of the 100 points shown in the level curve for $p = 10$ in the top (respectively, bottom) graph of Figure 3.8. As expected, we see that as the number of supply points used for quantization increases, the normalized quantization load improves; that is, as p increases, NQL_D approaches 1 from above. Comparing the top graph to the bottom graph, we can see that the variation in NQL_D decreases as the task set X increases in size from $n = 100$ to $n = 1000$.

Figures similar to Figure 3.8 for the remaining input distributions exhibit similar characteristics (Figures 3.9-3.13).

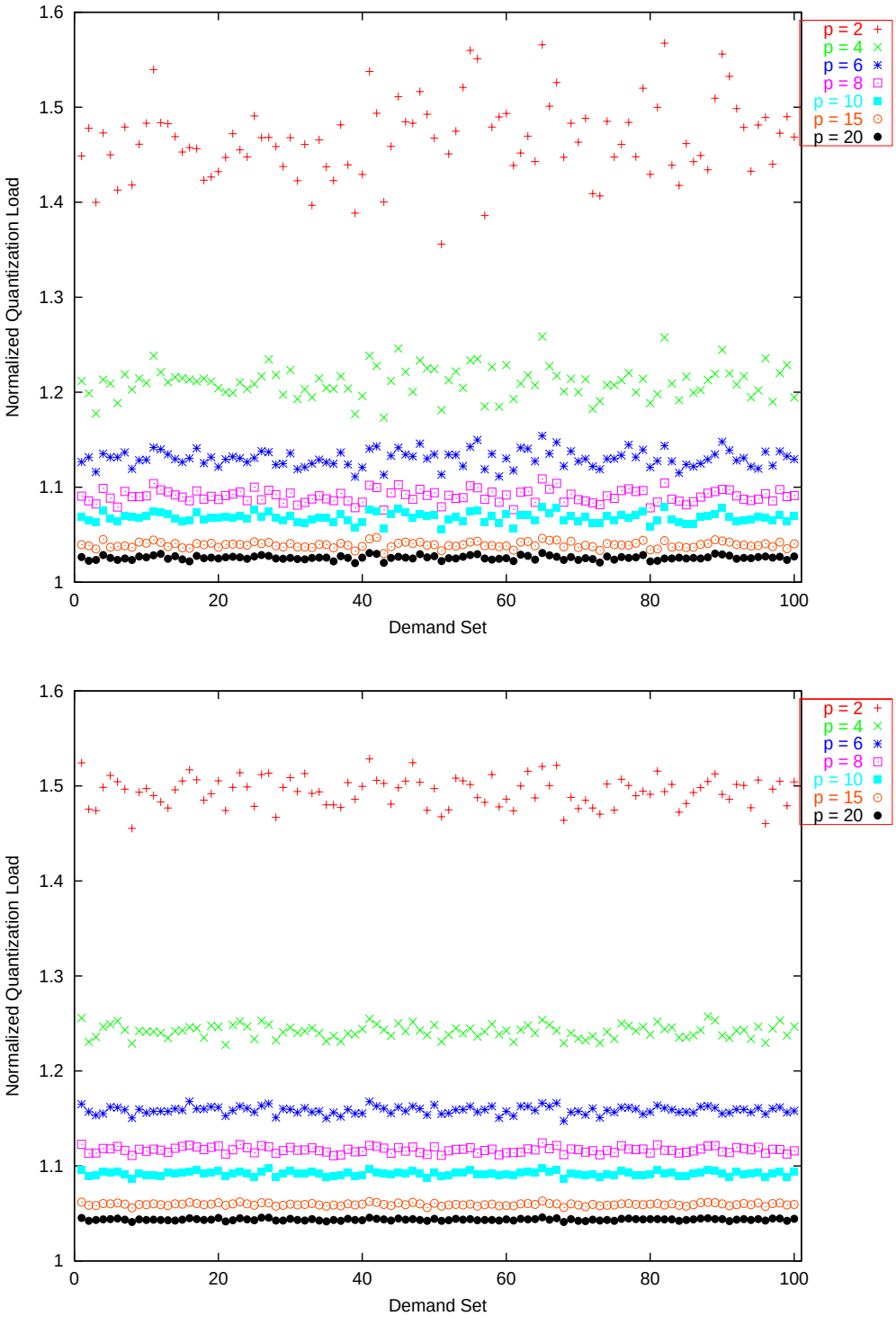


Figure 3.8: Uniform input: Level curves in p of the normalized quantization load, for 100 individual demand sets. Top: $n = 100$. Bottom: $n = 1000$.

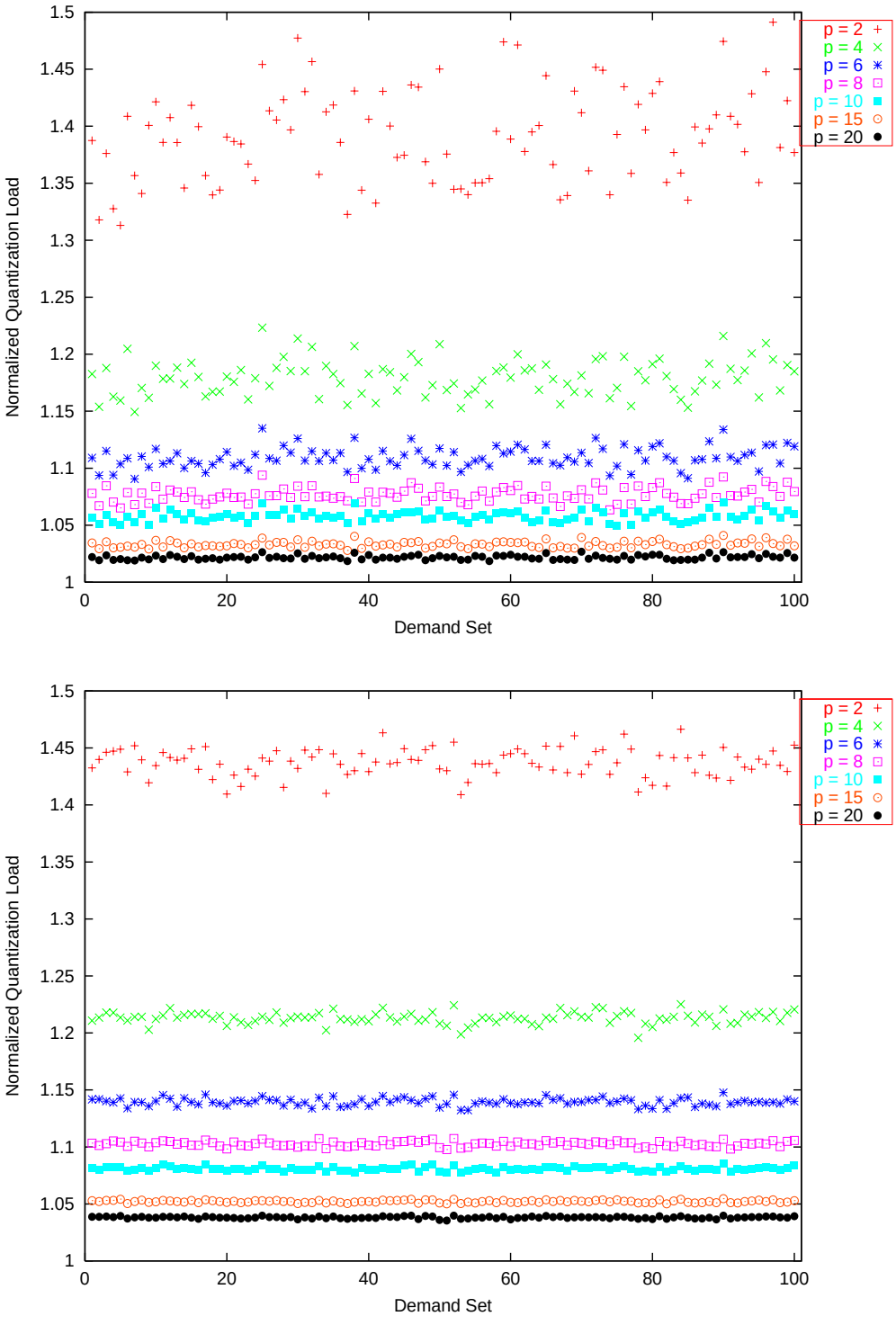


Figure 3.9: Triangle input: Level curves in p of the normalized quantization load, for 100 individual task sets. Top: $n = 100$. Bottom: $n = 1000$.

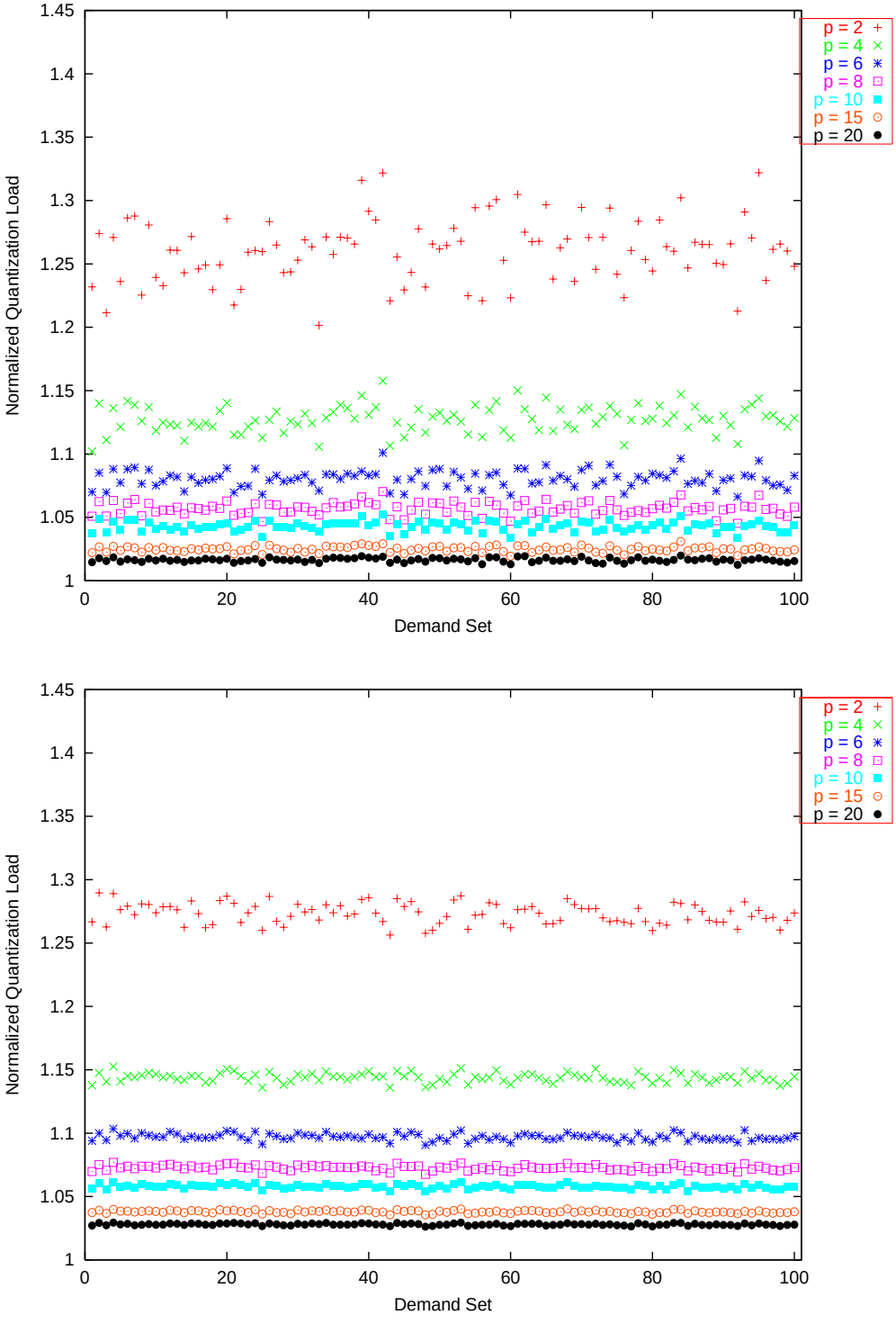


Figure 3.10: Increasing input: Level curves in p of the normalized quantization load, for 100 individual demand sets. Top: $n = 100$. Bottom: $n = 1000$.

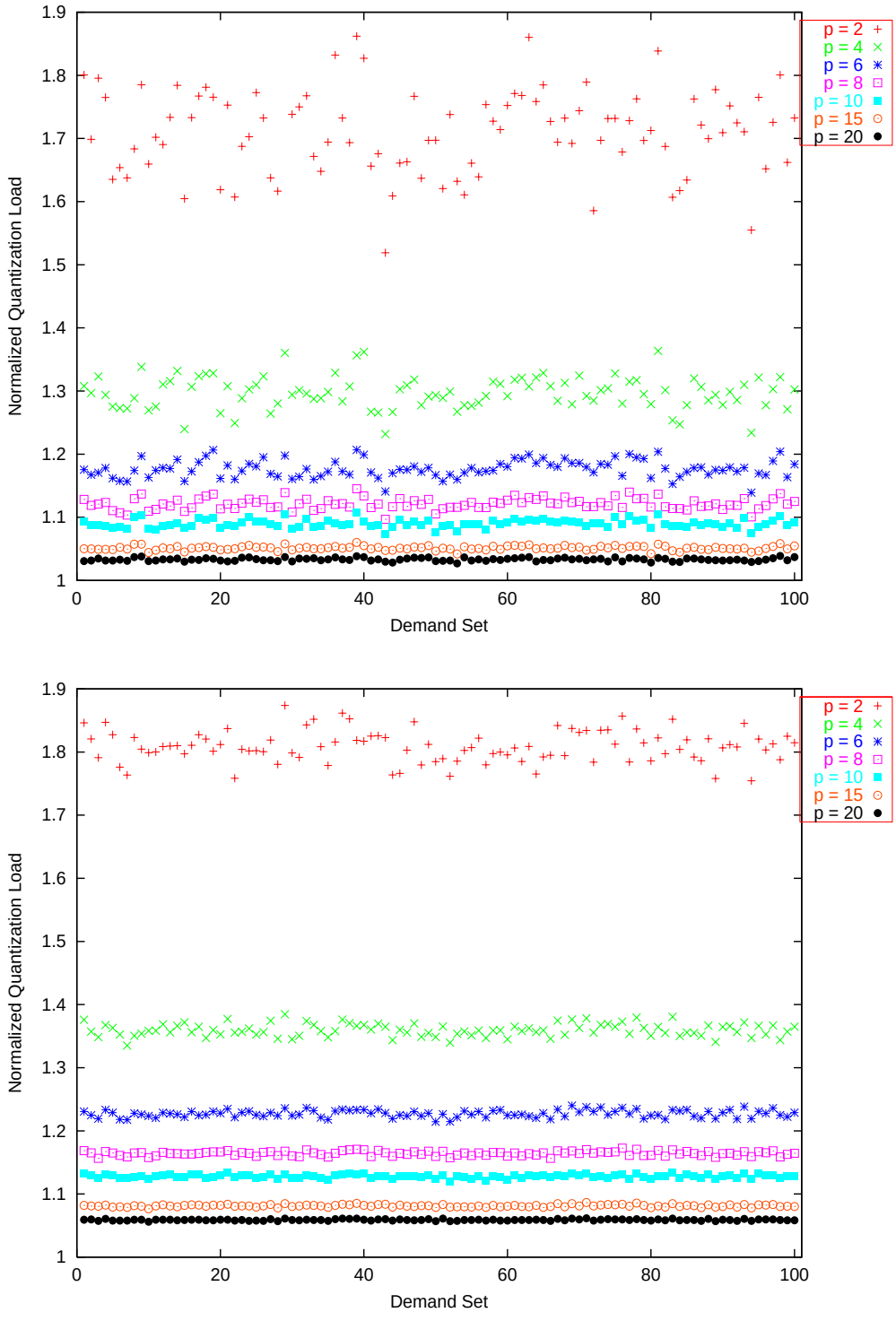


Figure 3.11: Decreasing input: Level curves in p of the normalized quantization load, for 100 individual demand sets. Top: $n = 100$. Bottom: $n = 1000$.

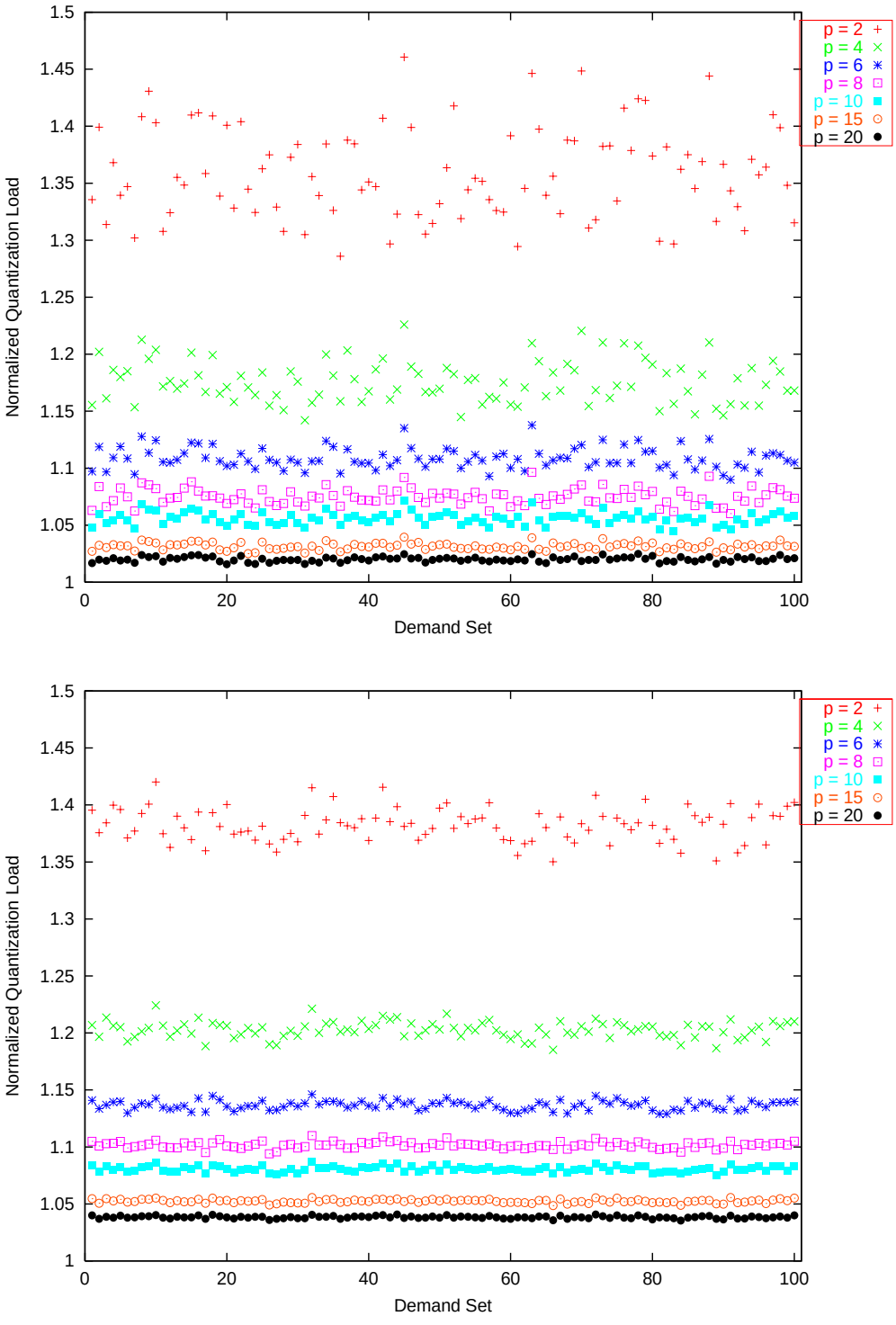


Figure 3.12: Unimodal input: Level curves in p of the normalized quantization load, for 100 individual demand sets. Top: $n = 100$. Bottom: $n = 1000$.

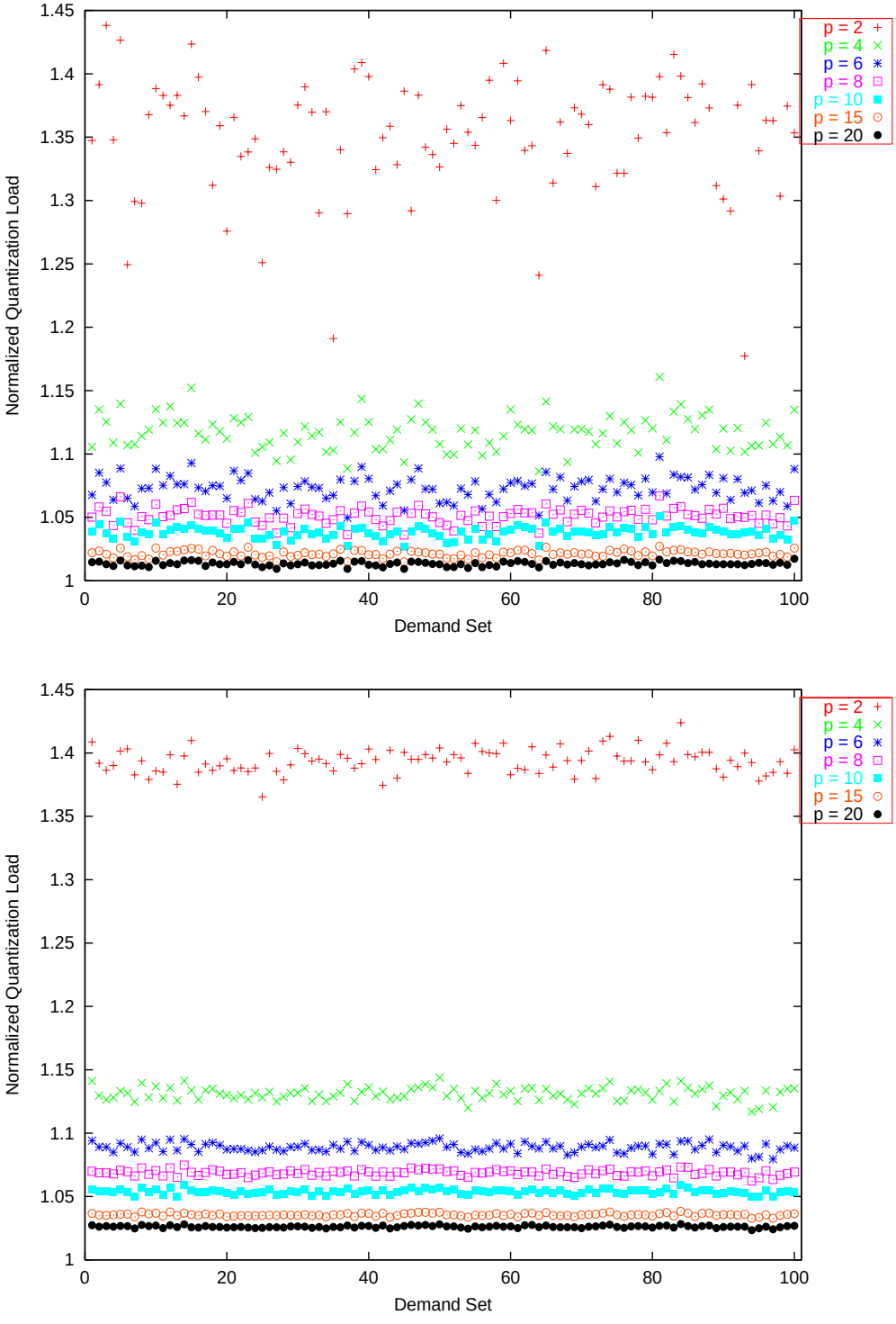


Figure 3.13: Bimodal input: Level curves in p of the normalized quantization load, for 100 individual demand sets. Top: $n = 100$. Bottom: $n = 1000$.

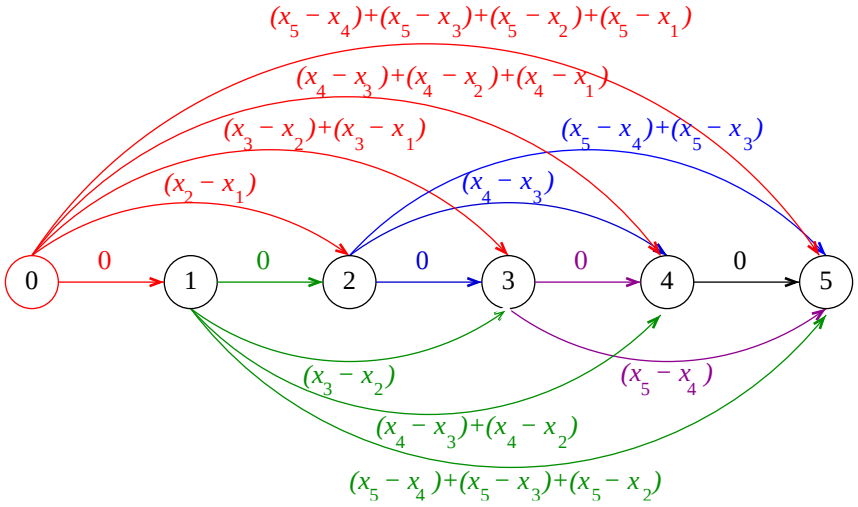


Figure 3.14: Graph representation of an instance of DPM1 with $n = 5$.

3.4 Graph representation of DPM1: A faster algorithm

Let $G = (V, E)$ be a weighted, complete, directed acyclic graph (DAG), with the vertex set $V = 0, 1, \cdots, n$ and arc weights $w(i, j)$ for arc (i, j) from vertex i to vertex j, $0 \leq i < j$. Solving problem DPM1 is equivalent to finding a minimum weight p-link path from vertex 0 to vertex n in G, a special case of the constrained shortest path problem (which is listed as problem ND30 in [12]) (see also [28]). Further, the arc weights in the graph representation of DPM1 can be shown to have the concave Monge property, allowing a solution in time $O\left(n\sqrt{p \log n}\right)$ [1], instead of the $O(n^2 p)$ required by algorithm Quantize. Figure 3.14 shows the graph for an instance of DPM1 with $n = 5$.

For each demand x_i there is one node i. In addition we create a dummy node 0. Let $w(i, k)$ be the arc weight from vertex i to vertex k. The value of $w(i, k)$ represents the cost of mapping demand points $i + 1$, $i + 2$, ...,

k to demand point k. We therefore assign $w(i,k)$ the following value:

$$w(i,k) = \begin{cases} 0, & k = i+1 \\ (k-i-1)x_k - \sum_{j=i+1}^{k-1} x_j, & k > i+1 \end{cases}$$

The objective is to find the minimum weight path from vertex 0 to vertex n that has exactly p arcs. A path t in a DAG is a sequence of arcs:

$$t \ : \ (i_0, i_1), (i_1, i_2), \ldots, (i_{r-1}, i_r)$$

Path t is called a (0-n)-path if $i_0 = 0$ and $i_r = n$; that is, t begins at vertex 0 and ends at vertex n. The weight of path t is simply:

$$w(t) = w(i_0, i_1) + w(i_1, i_2) + \cdots + w(i_{r-1}, i_r)$$

Any p-link path $t = (i_0, i_1), (i_1, i_2), \ldots, (i_{p-1}, i_p)$ with $i_0 = 0$ and $i_p = n$ is a feasible solution for DPM1. The interpretation of this path is as follows. The demand points corresponding to nodes $i_1, i_2, \ldots, i_p$ are designated as supply points. For example, in Figure 3.14, suppose $p = 3$, and let (0,2,4,5) be a feasible 3-link path. Then the corresponding feasible solution for DPM1 is $z_1 = x_2$, $z_2 = x_4$, and $z_3 = x_5$. The sum of the arc weights for this path equals the objective function value for the implied mapping for the corresponding solution $S = \{z_1, z_2, z_3\}$, namely:

$$\begin{aligned} w(t) &= w(0,2) + w(2,4) + w(4,5) \\ &= (x_2 - x_1) + (x_4 - x_3) + 0 \end{aligned}$$

Let $G = (V, E)$ be a weighted, complete DAG, with $V = v_0, v_1, \cdots, v_n$. G satisfies the concave Monge condition if

$$w(i, j) + w(i + 1, j + 1) \leq w(i, j + 1) + w(i + 1, j) \tag{3.8}$$

holds for all $0 < i + 1 < j < n$. It is easily shown that the arc weights for the graph representation of DPM1 obey this condition. We evaluate the left hand side (LHS) and right hand side (RHS) of Equation (3.8), and then show that RHS $-$ LHS ≥ 0.

$$\begin{aligned} \text{LHS} &= w(i, j) + w(i + 1, j + 1) \\ &= (j - i - 1)x_j - \sum_{m=i+1}^{j-1} x_m + (j - i - 1)x_{j+1} - \sum_{m=i+2}^{j} x_m \end{aligned}$$

$$\begin{aligned} \text{RHS} &= w(i, j + 1) + w(i + 1, j) \\ &= (j - i)x_{j+1} - \sum_{m=i+1}^{j} x_m + (j - i - 2)x_j - \sum_{m=i+2}^{j-1} x_m \end{aligned}$$

$$\begin{aligned} \text{RHS} - \text{LHS} &= (j - i - (j - i - 1))x_{j+1} + (j - i - 2 - (j - i - 1))x_j \\ &\qquad + \left(\sum_{m=i+1}^{j-1} x_m - \sum_{m=i+1}^{j} x_m + \sum_{m=i+2}^{j} x_m - \sum_{m=i+2}^{j-1} x_m \right) \\ &= x_{j+1} - x_j + (-x_j + x_j) \\ &= x_{j+1} - x_j \end{aligned}$$

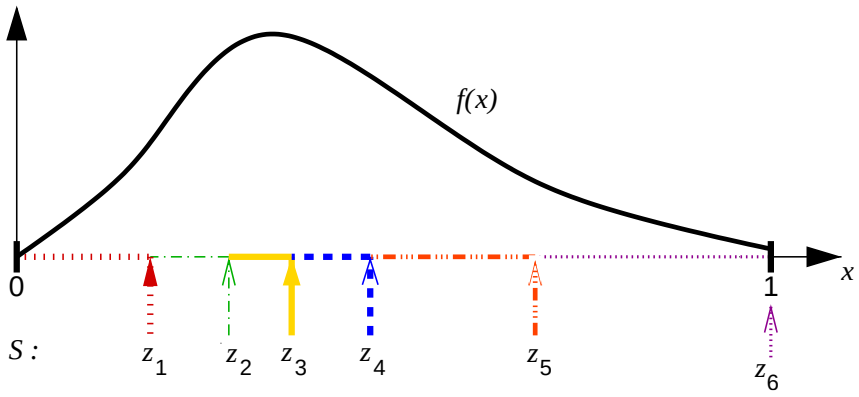


Figure 3.15: Sample mapping from the domain of $f(x)$ to a solution set of 6 supply points.

Recalling that $x_1 \leq x_2 \leq \cdots \leq x_n$, then we have

$$\text{RHS} - \text{LHS} = x_{j+1} - x_j \geq 0\,.$$

Thus the arc weights of the DPM1 graph representation obey the concave Monge property, and DPM1 can be solved in time $O\left(n\sqrt{p \log n}\right)$ using the algorithm in [1].

3.5 The stochastic directional p-median problem on the real line

3.5.1 Statement of the problem: SDPM1

We now consider a variation of the directional p-median problem illustrated in Figure 3.15. Let $f(x)$ and $F(x)$ be the probability density function and cumulative distribution function, respectively, representing the population of demand points, with domain wholly contained within $(0, 1)$. Let μ be the mean of $f(x)$ and let $b \leq 1$ be the least upper bound on the domain of $f(x)$. A set $S = \{z_1, \ldots, z_p\}$, $0 < z_1 < z_2 < \cdots < z_p < 1$, is a **feasible solution** of $f(x)$

if and only if $b \leq z_p$. For notational convenience, we set $z_0 = 0$. Notice that each z_j is unique. Associated with a feasible solution is an **implied mapping** from the domain of $f(x)$ into S, where $x \rightarrow z_j$ if and only if $z_{j-1} < x \leq z_j$. We may also write the implied mapping as $(x_{lower}, x_{upper}] \rightarrow z_j$, where $x_{lower} = z_{j-1}$ and $x_{upper} = z_j$.

Problem 3.2 (SDPM1) Given $f(x)$, $F(x)$, and μ as defined above, find a feasible solution of p quantized supply points z_j, $j = 1, \ldots, p$, such that the following objective function is minimized:

$$
\begin{aligned}
g_S(z_1, \ldots, z_p) &= \sum_{j=1}^{p} \left(\int_{z_{j-1}}^{z_j} (z_j - x)\, f(x)\, dx \right) && (3.9) \\
&= \sum_{j=1}^{p} \left(\int_{z_{j-1}}^{z_j} z_j\, f(x)\, dx \right) - \sum_{j=1}^{p} \left(\int_{z_{j-1}}^{z_j} x\, f(x)\, dx \right) && (3.10) \\
&= \sum_{j=1}^{p} \left(z_j \int_{z_{j-1}}^{z_j} f(x)\, dx \right) - \mu && (3.11) \\
&= q_S(z_1, \ldots, z_p) - \mu && (3.12)
\end{aligned}
$$

Notice that $g_S(z_1 \ldots z_p)$ is the *average* penalty per demand of excess resources used by the quantized set above that requested by the original demand set. The second term of Equation (3.12), μ, is the *average* load requested by a demand point, while the first term, $q_S(z_1, \ldots, z_p)$, is the *average* quantization load, that is, the average load across the set of quantized demands. In contrast, in the deterministic input case given in Equation (3.3), $g_D(z_1, \ldots, z_p) = q_D(z_1, \ldots, z_p) - \rho_X$ is the *total* penalty under quantization for a particular demand set X, not the average, and $q_D(z_1, \ldots, z_p)$ (respectively, ρ_X) is the total load for the quantized demand set (respectively, for the original demand set). We reach this conclusion mathematically by dividing

Equation (3.2) by n and taking the limit as n goes to infinity:

$$\begin{aligned}
\lim_{n\to\infty} \frac{g_D(z_1,\ldots,z_p)}{n} &= \lim_{n\to\infty}\left(\frac{1}{n}\left(\sum_{j=1}^{p}(n_j z_j)-\rho_X\right)\right) \\
&= \lim_{n\to\infty}\left(\sum_{j=1}^{p}\left(\frac{n_j}{n}z_j\right)\right)-\lim_{n\to\infty}\left(\frac{\rho_X}{n}\right) \\
&= \sum_{j=1}^{p}\left(z_j \lim_{n\to\infty}\left(\frac{n_j}{n}\right)\right)-\mu
\end{aligned}$$

Notice that the limit of n_j/n as n goes to infinity equals the proportion of x_i's that fall within the interval (z_{j-1}, z_j), or $\int_{z_{j-1}}^{z_j} f(x)\,dx$. Thus we have:

$$\begin{aligned}
\lim_{n\to\infty} \frac{g_D(z_1,\ldots,z_p)}{n} &= \sum_{j=1}^{p}\left(z_j \int_{z_{j-1}}^{z_j} f(x)\,dx\right)-\mu \\
&= g_S(z_1,\ldots,z_p)
\end{aligned}$$

We can find an expression for Normalized Quantization Load for stochastic input, NQL_S, by taking the limit of Equation (3.5) as n goes to infinity (notice that in going from Equation (3.13) to (3.14) below, we multiply the numerator and denominator by $1/n$):

$$\begin{aligned}
NQL_S &= \lim_{n\to\infty} \frac{\sum_{j=1}^{p}(n_j z_j)}{\rho_X} && (3.13)\\
&= \lim_{n\to\infty} \frac{\sum_{j=1}^{p}\left(\frac{n_j}{n}z_j\right)}{\frac{\rho_X}{n}} && (3.14)\\
&= \frac{\sum_{j=1}^{p}\left(z_j \lim_{n\to\infty}\left(\frac{n_j}{n}\right)\right)}{\lim_{n\to\infty}\left(\frac{\rho_X}{n}\right)} && (3.15)\\
&= \frac{\sum_{j=1}^{p}\left(z_j \int_{z_{j-1}}^{z_j} f(x)\,dx\right)}{\mu} && (3.16)\\
&= \frac{q_S(z_1,\ldots,z_p)}{\mu} && (3.17)
\end{aligned}$$

Because μ is a constant for a given $f(x)$, both the objective function $g_S(z_1, \ldots, z_p)$ and the Normalized Quantization Load NQL_S are minimized whenever the average quantization load $q_S(z_1, \ldots, z_p)$ is minimized. The following lemma is analogous to the fact that, in the deterministic case, the largest supply point in an optimal quantization set must equal the largest demand point x_n.

Lemma 3.5 Let $f(x)$, $F(x)$, and b be defined as above. Let $S = \{z_1, \ldots, z_p\}$, $0 < z_1 < z_2 < \cdots < z_p < 1$, be an optimal solution of $f(x)$. Then $z_p = b$.

Proof. By contradiction. Suppose $z_p \neq b$. From the definition of a feasible solution, we know $b \leq z_p$; thus $b < z_p$. The values currently mapped to z_p lie in the interval $(z_{p-1}, b]$. Moving z_p down to b will reduce the objective function by a non-negligible amount equal to $(z_p - b) \int_{z_{p-1}}^{b} f(x)\, dx$. This contradicts the optimality of S. Thus $z_p = b$. ■

3.5.2 Optimal solution through nonlinear programming

For a given cumulative distribution function $F(x)$ and given values of p and b, we can optimally solve problem SDPM1 using the method described in this section, whenever $F(x)$ is (1) twice differentiable and (2) not piecewise defined, over the entire domain of $F(x)$. In Section 3.5.3 we present an approximate solution for instances of SDPM1 for which $F(x)$ fails to have these two properties.

Rewriting $g_S(z_1, \ldots, z_p)$ from Equation (3.11), we have the following

optimization problem:

$$\begin{aligned} \text{Minimize } g_S(z_1,\ldots,z_p) &= \sum_{j=1}^{p} \left(z_j \left(F(z_j) - F(z_{j-1})\right)\right) - \mu \\ \text{subject to :} &\quad 0 < z_1 < z_2 < \cdots < z_{p-1} < z_p = b \end{aligned}$$

When $F(x)$ is twice differentiable and not piecewise defined, $f(x)$ and $f'(x)$ are also not piecewise defined. Specifically, for each of $F(x)$, $f(x)$, and $f'(x)$, it is possible to write the function as a single closed form expression over its entire domain, a necessary property for applying the following method: locate a critical point of g_S and then verify that the point is a minimum.

To find a critical point, we set the first order partial derivatives of g_S with respect to z_j, $j = 1,\ldots,p-1$, equal to zero, yielding a set of $p-1$ simultaneous differential equations in $p-1$ unknowns. The highest service level z_p is known; from Lemma 3.5we know $z_p = b$. It will then be possible to solve for each z_j, $j = 2,\ldots,$ p, in terms of z_1 only. Since $z_p = b$, we can find z_1. Through back-substitution we can then obtain the remaining values for z_j, $j = 2,\ldots,$ $p-1$.

Taking the partial derivative of g_S with respect to z_j, $j = 1,\ldots,p-1$, we have:

$$\begin{aligned} \frac{\partial g_S}{\partial z_j} &= z_j \frac{\partial F(z_j)}{\partial z_j} + (F(z_j) - F(z_{j-1})) - z_{j+1}\frac{\partial F(z_j)}{\partial z_j} && (3.18) \\ &= (z_j - z_{j+1})\frac{\partial F(z_j)}{\partial z_j} + F(z_j) - F(z_{j-1}) && (3.19) \\ &= (z_j - z_{j+1}) f(z_j) + F(z_j) - F(z_{j-1}) && (3.20) \end{aligned}$$

From the equation $\frac{\partial g_S}{\partial z_j} = 0$, $j = 1,\ldots,p-1$, we can solve for z_{j+1} in

terms of z_j and z_{j-1}:

$$z_{j+1} = z_j + \frac{F(z_j) - F(z_{j-1})}{f(z_j)} \tag{3.21}$$

Since $z_0 = 0$, then $F(z_0) = 0$. For the equation corresponding to $\frac{\partial g_S}{\partial z_1} = 0$, we have:

$$z_2 = z_1 + \frac{F(z_1)}{f(z_1)} \tag{3.22}$$

Thus we have z_2 in terms of z_1 only. For the equation corresponding to $\frac{\partial g_S}{\partial z_2} = 0$, we have:

$$z_3 = z_2 + \frac{F(z_2) - F(z_1)}{f(z_2)} \tag{3.23}$$

Using Equation (3.22) to substitute for z_2 in Equation (3.23) above gives an expression for z_3 in terms of z_1 only. For the equation corresponding to $\frac{\partial g_S}{\partial z_3} = 0$, we have:

$$z_4 = z_3 + \frac{F(z_3) - F(z_2)}{f(z_3)} \tag{3.24}$$

Since we already have both z_3 and z_2 in terms of z_1 only, we can use substitution to get z_4 in terms of z_1 only. In general, we can obtain an expression for z_{j+1} in terms of z_1 only, after using substitution in the equation corresponding to $\frac{\partial g_S}{\partial z_j} = 0$. The final equation, corresponding to $\frac{\partial g_S}{\partial z_{p-1}} = 0$, is:

$$b \;=\; z_p \;=\; z_{p-1} + \frac{F(z_{p-1}) - F(z_{p-2})}{f(z_{p-1})}$$

After substitution, the left-hand side of this equation is the constant b, and the right-hand side is a function of z_1. Thus we can solve for z_1. All

other values of z_j, $j = 2, \ldots, p-1$, can be obtained once z_1 is known.

Notice that the feasible region, defined by $0 < z_1 < z_2 < \ldots < z_{p-1} < z_p = b$, is a convex set. If $F(x)$ is a convex function, then g_S is also convex, and the critical point $(z_1, z_2, \ldots, z_{p-1})$ will be a global minimum. Otherwise, the critical point $(z_1, z_2, \ldots, z_{p-1})$ is a minimum if and only if the Hessian matrix of second partial derivatives of g_S is positive definite. Since the Hessian for g_S turns out to be a symmetric tridiagonal matrix, it can be shown to be positive definite (or not) in time $O(p^2)$ [8].

Example: Solution for the uniform input distribution. Due to the simplicity of the uniform distribution, namely $f(x) = 1$ and $F(x) = x$, it is possible to solve for the optimal values of $z_1, \ldots, z_{p-1}$ without specifying a particular value for p. The domain of the uniform distribution is $(0, 1)$, thus from Lemma 3.5 we have $z_p = 1$. Using Equation (3.21) we have:

$$\begin{aligned} z_{j+1} &= z_j + \frac{z_j - z_{j-1}}{1} \\ &= 2z_j - z_{j-1} \end{aligned}$$

Recalling $z_0 = 0$, the first equation (corresponding to $\frac{\partial g_S}{\partial z_1} = 0$) yields: $z_2 = 2z_1$. From the second equation we have: $z_3 = 2z_2 - z_1 = 3z_1$; from the third: $z_4 = 2z_3 - z_2 = 4z_1$; and so on, up to the $(p-1)^{\text{st}}$ equation: $z_p = pz_1$. In general, $z_j = jz_1$ for $j = 2, \ldots, p$. Using the additional information that $z_p = 1$, we have that $z_p = pz_1 = 1$. Thus $z_1 = \frac{1}{p}$, and for $j = 2, \ldots, p-1$ we have $z_j = \frac{j}{p}$. This result confirms our intuition, which tells us that whenever demand points are scattered uniformly on $(0, 1)$, the best approach is to place the p supply points at equal intervals on $(0, 1)$.

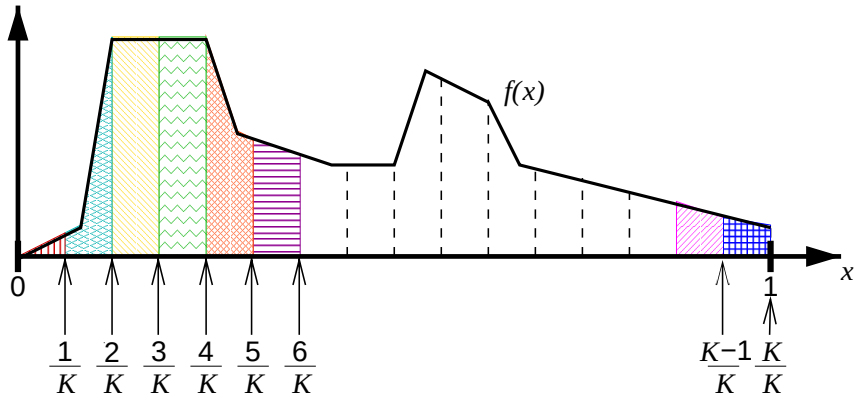


Figure 3.16: Forming a pdf approximation: The area under $f(x)$ over an interval is paired with the right-hand endpoint of the interval.

3.5.3 An approximate solution: Algorithm Quantize-Continuous

For any given cumulative distribution function $F(x)$ and given values of p and b, we can find an approximate solution to problem SDPM1 using the method described in this section and illustrated in Figure 3.16. This approximation is necessary whenever $F(x)$ is piecewise defined or fails to be twice differentiable; in addition, this approximation may be used whenever the complexity of $F(x)$ and $f(x)$ make the approach of Section 3.5.2 difficult.

It is possible to create a discrete approximation of the pdf and use an algorithm similar to Quantize to find an estimate of the optimal solution set S. That is, the new algorithm, called Quantize-Continuous, will find the optimal quantization set for a given approximation of a pdf. The better the pdf approximation, the closer the estimate will be to the true optimal solution for the pdf.

In particular, we can choose an integer $K > p$ and partition the interval $(0, 1)$ into K intervals $(\frac{i-1}{K}, \frac{i}{K})$, $i = 1, \ldots, K$. The right-hand endpoint

of the i^{th} interval is $e_i = \frac{i}{K}$; we associate with e_i a discrete point mass density

$$m_i = \int_{\frac{i-1}{k}}^{\frac{i}{k}} f(x)\,dx\,.$$

These K ordered pairs $\{(e_i, m_i)\}$ form the approximation of $f(x)$ that serves as input to the algorithm Quantize-Continuous, which selects the z_j's of the optimal solution set S from among the K endpoint values $\{e_i\}$.

Quantize-Continuous differs from Quantize in several ways. First, the input to Quantize-Continuous is the collection of K ordered pairs $\{(e_i, m_i)\}$, while the input to Quantize is the set of demand points X, containing n values of x_i. Second, the tables `Diff` and `Cumul` are replaced by the $K \times K$ tables `Sum` and `Prod`:

$$\begin{aligned} \texttt{Sum}[i][j] &= \sum_{s=i}^{j} \texttt{m}[s]\ ,\ i \le j \\ \texttt{Prod}[i][j] &= \texttt{e}[i] \cdot \texttt{Sum}[j][i]\ ,\ j \le i \end{aligned}$$

Lastly, Quantize-Continuous minimizes the average quantization load $q_S(z_1, \ldots, z_p)$ (and holds these values in a $K \times p$ table called `AQL`), whereas Quantize minimizes the total quantization load $q_D(z_1, \ldots, z_p)$ (and holds the values of $q_D(z_1, \ldots, z_p) - \rho_X = g_D(z_1, \ldots, z_p)$ in the `Opt` table). Apart from these differences, the two algorithms are very similar, in that the same code used in Quantize to build `Opt` (using `Cumul`) is exactly the same code used in Quantize-Continuous to build `AQL` (using `Prod` in the place of `Cumul`). Quantize-Continuous runs in time $O(K^2 p)$ and Quantize runs in time $O(n^2 p)$; these time complexities are identical, since K and n simply represent the size of the input. Table 3.3 summarizes the main data structures used by Quantize-Continuous.

name	size	defined	description of entry i, j
`Sum`	$K \times K$	$i \leq j$	the sum: $\sum_{s=i}^{j}$ `m[`s`]`
`Prod`	$K \times K$	$j \leq i$	the product: `e[`i`]`$\times$`Sum[`j`][`i`]`
`AQL`	$K \times p$	$j \leq i$	portion of g_S due to an optimal choice of $p = j$ from $(e_1, m_1) \ldots (e_i, m_i)$
`Prev`	$K \times p$	$j \leq i$, $j \neq 1$	index into `e` array of z_{j-1}, if $z_j =$`e[`i`]`

Table 3.3: Tables used in the Quantize-Continuous algorithm.

Note that the entry `AQL[`i`][`j`]` holds the minimum value of q_S for a subset of the larger problem instance of SDPM1 that we wish to solve; namely, `AQL[`i`][`j`]` is the portion of $q_S(z_1, \ldots, z_p)$ that arises from optimally choosing j service levels to quantize the first i pairs (e_1, m_1) up to (e_i, m_i). (`AQL[`i`][`j`]` does not hold the minimum value of q_S for a smaller problem instance of SDPM1 in which the $K = i$ and $p = j$.) The pseudocode description of Quantize-Continuous can be found in Appendix B.

3.5.4 Performance of Quantize-Continuous on six input distributions

To evaluate the performance of Quantize-Continuous, we ran the algorithm on the six different input distributions described in Section 3.3 for a variety of values of K and p. In particular, we allowed K to take on the values $10, 15, 20, \ldots, 100$ and p the values from 2 to 50. However, in the graphs of Figures 3.17-3.19, we have chosen only to display level curves of p for $p = 5, 10, 15, 20, 25, 30$, and 35. These three figures contain two graphs apiece, corresponding to the six different input distributions as input to Quantize-Continuous. We have plotted the value of the normalized quantization load NQL_S on the y-axis corresponding to a particular value of K along the x-axis.

As expected, the level curves of p approach 1 as p increases. Notice also that, for a particular value of p, as K increases, the value of NQL_S decreases slightly and immediately settles down to a particular value. For example, in the bottom graph (triangle input) of Figure 3.17, the level curve of $p = 20$ settles down to a value of $NQL_S \approx 1.045$ as early as $K = 25$. Therefore by dividing the interval $(0, 1)$ into as few as 25 smaller intervals, we can adequately estimate the effect on required resources due to quantization into 20 supply points.

In the bottom graph (bimodal input) of Figure 3.19, the level curves of p don't appear to settle down as quickly; instead they possess an interesting sinusoidal shape. We therefore generated another graph (Figure 3.20) for the bimodal distribution, this time letting K take on the values $10, 15, 20, \ldots, 300$. From $K = 100$ to $K = 300$, the sinusoidal shape quickly decreases in amplitude and settles down to a particular value of NQL_S. The shape can be attributed to the endpoints of the K intervals adequately falling along the points of discontinuity of the pdf. The first peak in the bimodal distribution rises at $x = .25$ and falls at $x = .35$, and the second peak rises at $x = .65$ and falls at $x = .75$. When $K = 20, 40, 60, \ldots$, there are endpoints e_i that exactly equal .25, .35, .65, and .75; further, these K values correspond to the valleys (lower values of NQL_S) of the level curves of p.

Thus a probability density function $f(x)$ with discontinuities is better approximated (hence Quantize-Continuous performs better) whenever the K intervals are chosen such that the endpoints lie at the points of discontinuity. In fact, Quantize-Continuous does not require that the input pairs (e_i, m_i) be evenly spaced along the interval $(0, 1)$. Therefore whenever $f(x)$ has many discontinuities, the endpoints e_i may be particularly chosen to fall at the points of discontinuity to achieve better performance from Quantize-Continuous.

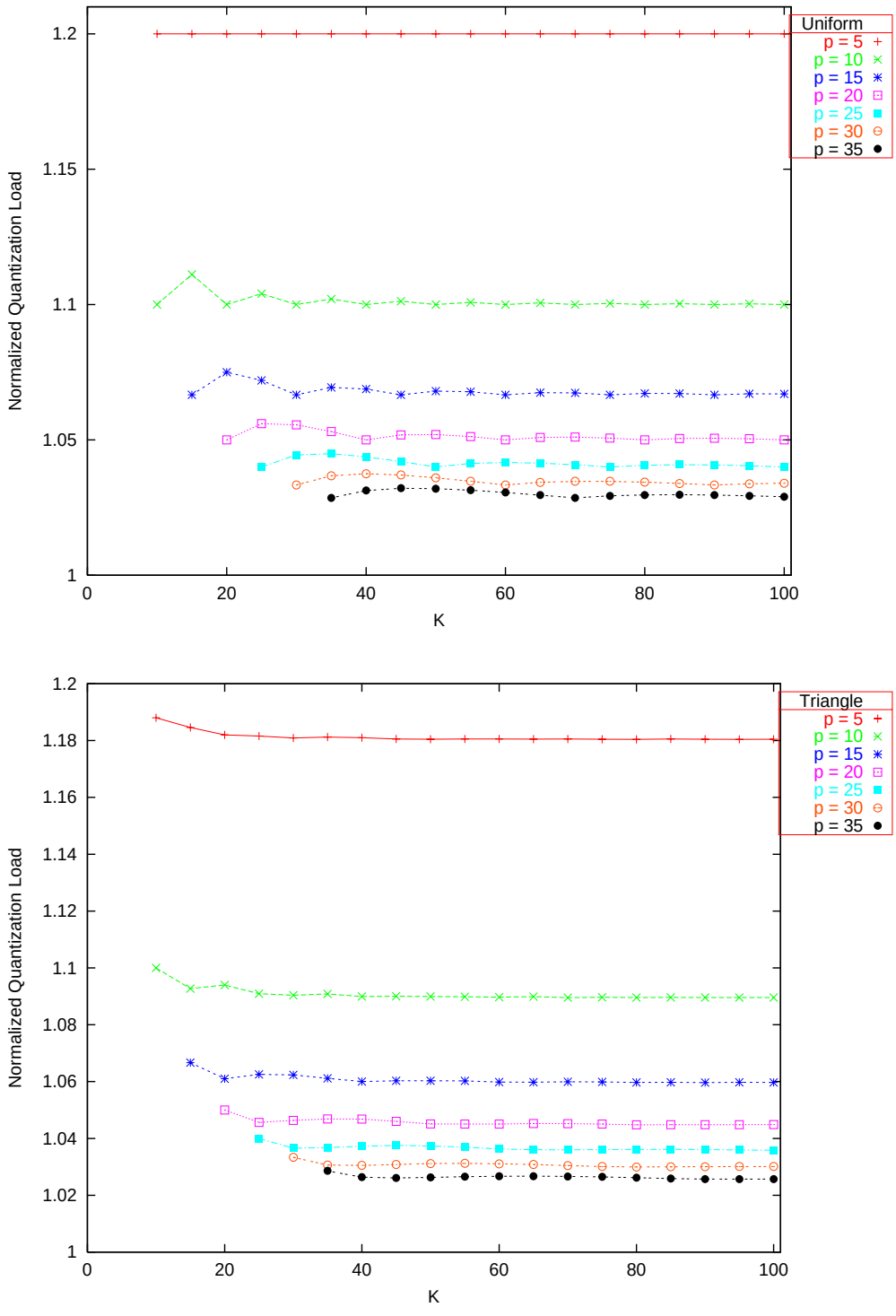


Figure 3.17: Level curves in p of the normalized quantization load, for $K = 10, 15, \ldots, 100$. Top: Uniform. Bottom: Triangle.

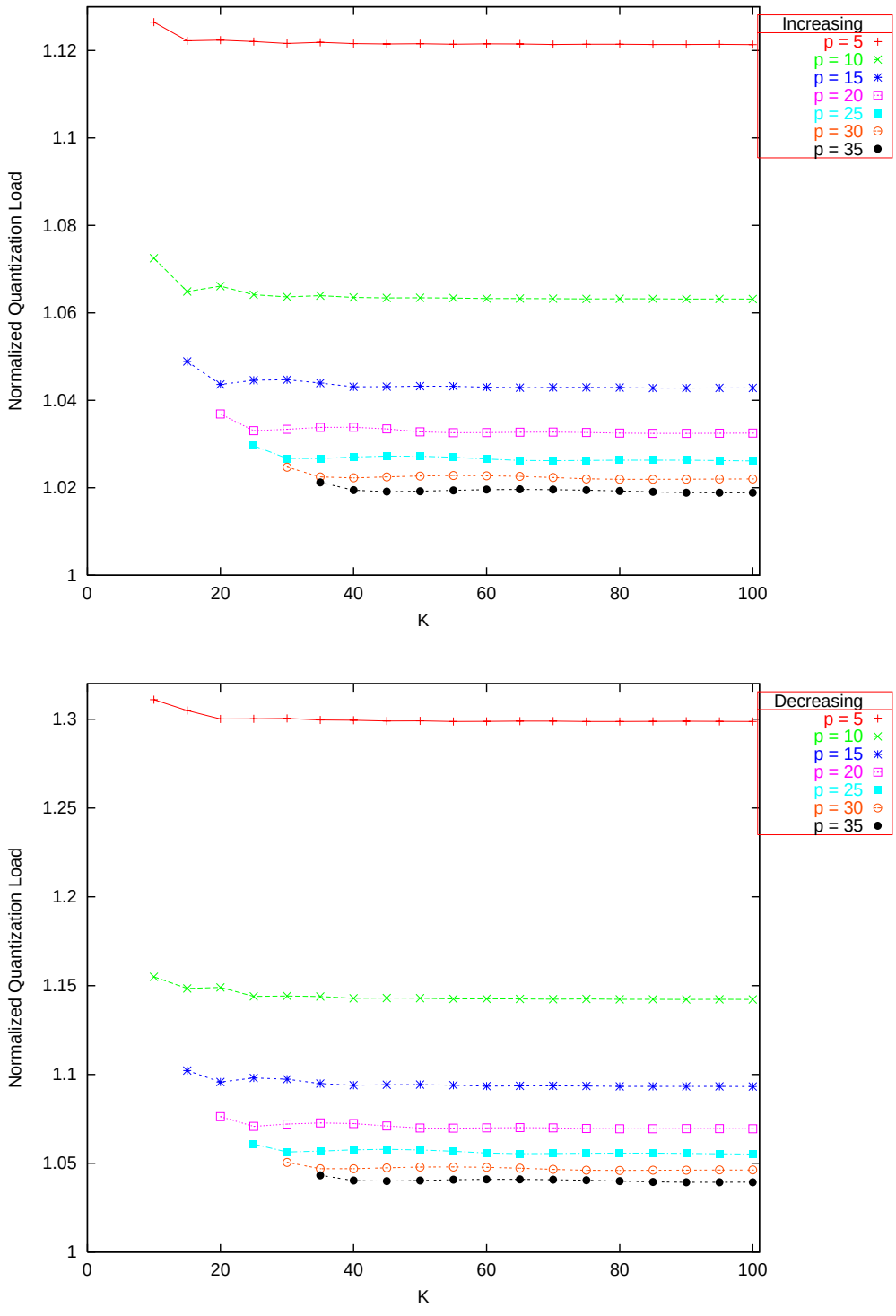


Figure 3.18: Level curves in p of the normalized quantization load, for $K = 10, 15, \ldots, 100$. Top: Increasing. Bottom: Decreasing.

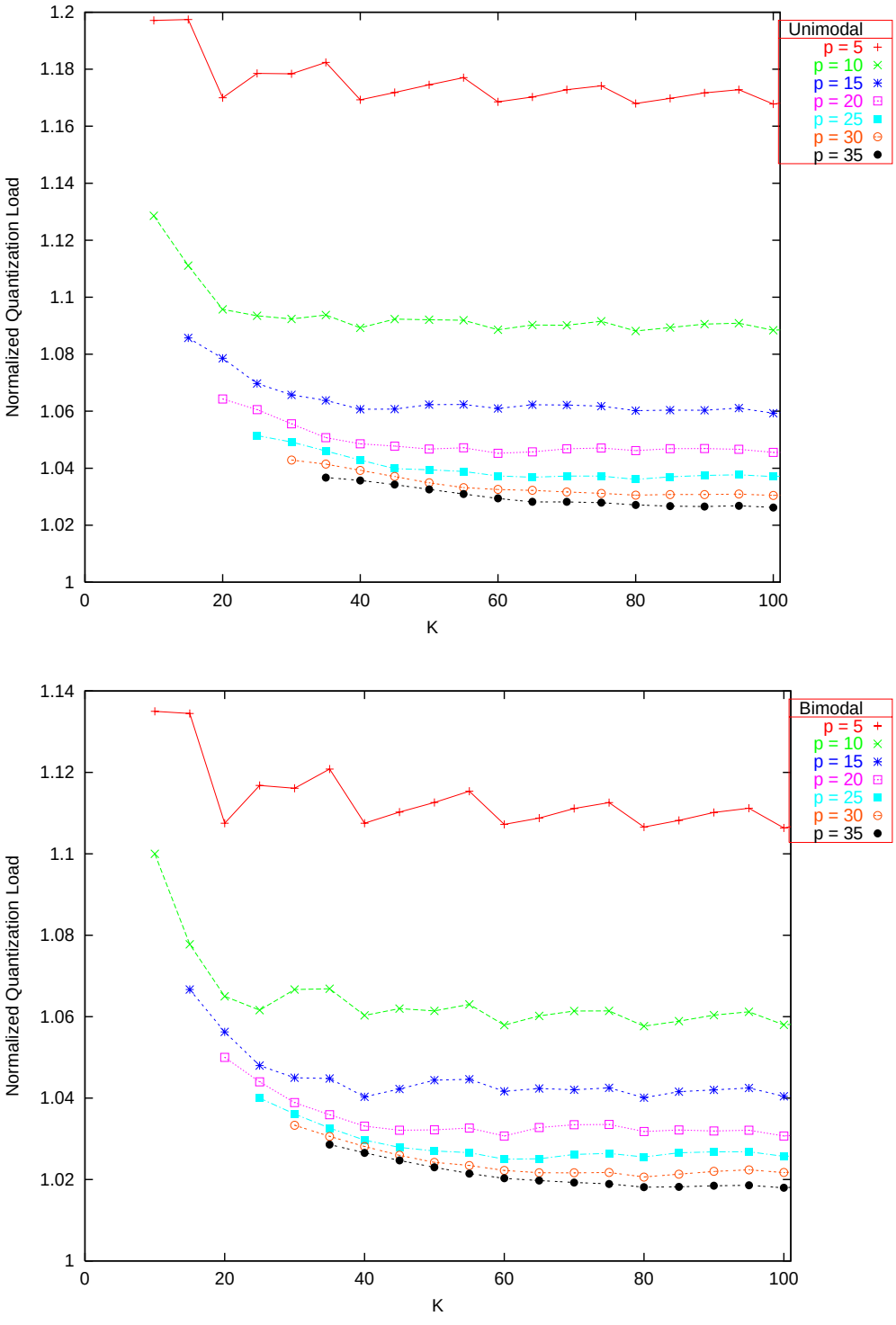


Figure 3.19: Level curves in p of the normalized quantization load, for $K = 10, 15, \ldots, 100$. Top: Unimodal. Bottom: Bimodal.

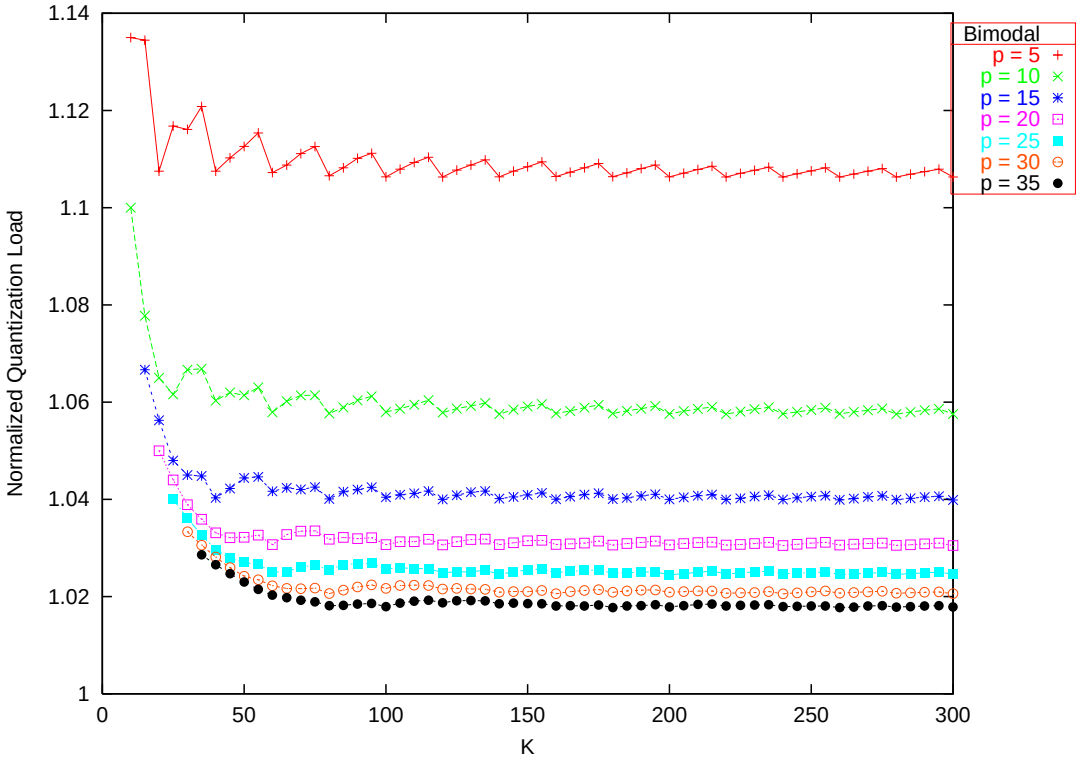


Figure 3.20: Bimodal: Level curves in p of the normalized quantization load, for $K = 10, 15, \ldots, 300$.

3.6 Application: Improved complexity for scheduling periodic tasks on a multiprocessor

In Section 2.2 we introduced the problem of scheduling n periodic tasks on m identical processors. The algorithm PD^2 is the fastest known algorithm that will create a feasible schedule for periodic task sets which satisfy the necessary and sufficient condition $\sum_{i=1}^{n} x_i \leq m$ [2]. Here, we give a faster algorithm (Q-PD^2) that schedules a quantized set of periodic tasks.

Recall from Section 2.2 that PD^2 uses the slot deadline associated with each subtask to prioritize subtasks for scheduling. Generally speaking, the next subtask of a particular task becomes **eligible** for processing only after the previous subtask's slot deadline has passed.[2] The online implementation of PD^2 presented in [2] has the following main phases:

(1) **Pre-Processing**. The algorithm inserts the initial eligible subtask of each of the n tasks into a heap H, which holds all subtasks currently eligible for processing.

(2) **Scheduling**. At each time slot:

   (a) **Selection**. The algorithm chooses a total of m eligible subtasks to process. It chooses subtasks according to most imminent deadline, and breaks ties in constant time.

   (b) **Update**. For each task corresponding to one of the m selected subtasks, the algorithm calculates the earliest time slot t at which the task's next subtask will become eligible. It then inserts this

[2]Depending on the value of a task's density $x_i = \frac{C_i}{D_i}$, the windows of time in which neighboring subtasks are eligible may, in fact, overlap by at most one slot. The reader is referred to [2] for details.

next subtask into a heap H_t according to its deadline. Since there are n tasks in all, the number of non-empty heaps is at most $n+1$.

PD^2 completes the Pre-Processing phase in time $O(n)$. During the Scheduling phase, PD^2 completes Selection in time $O(m \log n)$ and Update in time $O(m \log n)$. At any point in time, for each task, at most one subtask (the next one to be processed) is stored in one of several heaps: heap H if the subtask is currently eligible, or heap H_t if time slot t is the next earliest slot that the subtask will be eligible.

We propose modifying PD^2 to create a new algorithm called Quantized-PD^2, or Q-PD^2. We use the same priority definition as PD^2; that is, we use the same rules during the Selection phase to choose m eligible subtasks for processing. Taking advantage of the quantized input, we replace the collection of heaps with a set of p queues, one for each service level (supply point) z_j. During the Pre-Processing phase, the initial eligible subtask of each of the n tasks is inserted in arbitrary order into the queue corresponding to its assigned service level. Initially, all subtasks within a given queue have the same deadline and hence the same priority. Choosing the eligible head-of-line subtask with the highest priority from among the p queues can be done in time $O(1)$ (recall that, for the scheduling problem, p is a constant; it doesn't depend on n or m). The Selection phase can therefore be completed in time $O(m)$. Next, for each of the m selected subtasks, the Update phase involves:

(1) calculating the time `t_next` at which the task's next subtask will become eligible,

(2) calculating the priority of the next subtask at time `t_next`, and

(3) placing the next subtask at the end of its queue.

Each of these three actions for updating a single task requires time $O(1)$, so in total the Update phase requires time $O(m)$.

Therefore Q-PD^2 has a per slot time complexity of $O(m)$ as compared to $O(m \log n)$ for PD^2. The Pre-Processing phase time complexity remains unchanged at $O(n)$. The pseudocode description of Q-PD^2 is given in Appendix C.

3.7 Conclusion

In this chapter we considered two variants of the one-dimensional directional p-median problem. In the first, DPM1, the input is a finite set of demand points, and in the second, SDPM1, the input is the probability density function representing the population of demand points. We presented optimal solutions in each case. However, as we will see in the next chapter, the addition of even one more dimension causes directional p-median to become intractable.

Chapter 4

The complexity of directional p-median in two or more dimensions

In Chapter 3 we introduced the one-dimensional directional p-median problem (DPM1) and showed that it can be solved in time $O\left(n\sqrt{p\log n}\right)$ by restating the problem as a constrained shortest path problem. In this chapter we define the multidimensional counterpart of DPM1, the rectilinear c-directional, d-dimensional p-median problem, and prove it NP-complete. Chapter 5 presents an efficient heuristic for the multidimensional problem.

Recall that the one-dimensional (non-directional) p-median problem is solved in time O(pn) [13]. In two or more dimensions, the (non-directional) continuous p-median problem (Problem 1.1) is NP-complete under either the Euclidean or the rectilinear distance measure [17]. In the rectilinear case, only intersection points (defined in Section 1.2) need be considered as candidate sites

for supply points, which reduces the problem to a discrete p-median problem.

Similarly, the p-median problem under the *directional* rectilinear distance measure reduces to a discrete problem, and only *directional* intersection points need be considered as candidates. Recall from Section 1.3 that we define the c-directional, d-dimensional rectilinear distance (with $c \leq d$) from point $(p_1, \ldots, p_d)$ to $(q_1, \ldots, q_d)$ to be ∞ if $p_i > q_i$ for some $i \in \{1, \ldots, c\}$ and $\sum_{1 \leq i \leq d} |q_i - p_i|$ otherwise. Under a directional metric, a supply point must achieve or exceed the values of the first c coordinates of its assigned demand points. Notice that the distance matrix $[d_{ij}]$, where d_{ij} is the distance from point i to point j, is not symmetric, because if d_{ij} is finite and $d_{ij} > 0$, then d_{ji} is necessarily infinite. However, the directional distance measure does obey the triangle inequality ($d_{ij} \leq d_{ik} + d_{kj}$ for any three points i, j, and k), a characteristic which bodes well for the performance of heuristics for this problem [25].

The multidimensional directional p-median problem arises when attempting to proportion a network to satisfy the various Quality of Service (QoS) demands of many customers, where a particular level of QoS is described by $d \geq 2$ parameters. In two dimensions, for example, a service provider may characterize its service using the two parameters average transmission rate ρ and maximum burst size σ. Each customer requests a level of service described by an ordered pair (ρ_i, σ_i), and the collection of thousands of requests is a scatter plot in the (ρ, σ)-plane. In an effort to simplify network operations, the service provider may prefer to group similar requests into a single service level, in such a way that any given customer receives *at least* the amount requested. That is, a given demand point in the plane will be mapped to a supply point that is located above and to the right of it, corresponding to a c-directional, d-dimensional distance metric with $c = d = 2$.

The rest of the chapter is organized as follows. We begin in Section 4.1 with a precise statement of the decision version of the problem for $c = d = 2$ (DPM2). We show that DPM2 is NP-complete, which implies NP-completeness for all c, d satisfying $2 \leq c \leq d$. Sections 4.2 and 4.3 describe the two main components in the reduction from planar 3-SAT, circuits and clause configurations. We then present a sketch of the proof in Section 4.4. Sections 4.5.2 and 4.5.5 contain details of the proof.

4.1 Directional p-median in two dimensions

Our main result is that the (decision version of the) directional, rectilinear p-median problem is NP-complete in two dimensions. In the precise statement of the problem which follows, let $d_{dr}((x_i, y_i), (x_j, y_j))$ be the 2-directional rectilinear distance from point (x_i, y_i) to point (x_j, y_j). That is,

$$d_{dr}((x_i, y_i), (x_j, y_j)) = \begin{cases} x_j - x_i + y_j - y_i & \text{if } x_j \geq x_i \text{ and } y_j \geq y_i, \\ \infty & \text{otherwise} \end{cases}$$

Problem 4.1 (DPM2) Given a set $X = \{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\}$ of points in the plane, an integer p, and a bound B, does there exist a set

$$S = \{(z_1, t_1), (z_2, t_2), \ldots, (z_p, t_p)\}$$

of p points such that

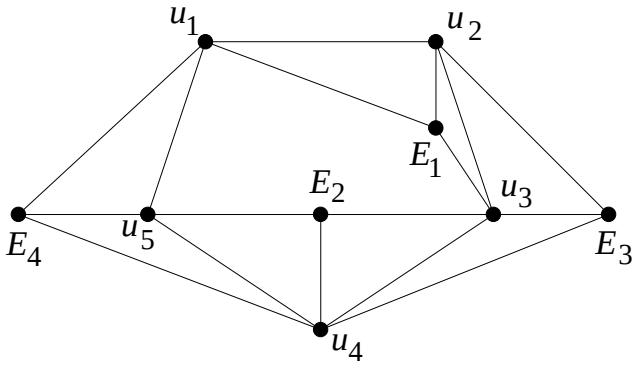
$$\sum_{i=1}^{n} \min_{1 \leq j \leq p} \{d_{dr}((x_i, y_i), (z_j, t_j))\} \leq B \, ?$$

Our reduction is from planar 3-SAT, the non-polar version (see [18]),

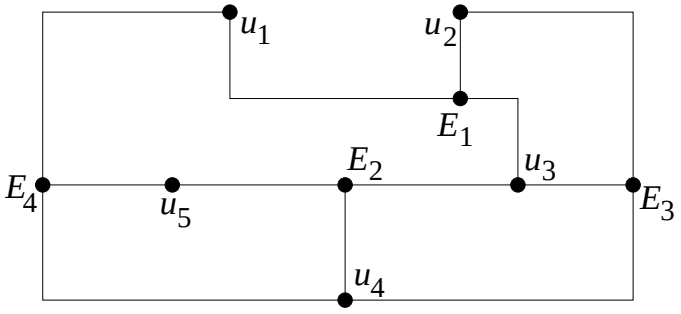
first proved NP-complete by Lichtenstein [16]. An instance of planar 3-SAT consists of a conjunctive normal form (CNF) formula with three literals per clause, each literal being either a variable or its negation. In addition, the graph representation of this formula must be planar. To build the graph representation, we create a vertex for each variable and a vertex for each clause. We add edges to connect variable v's vertex to the vertex of any clause in which v or its negation appears. The graph must remain planar even if we add a cycle that traverses the variable vertices in some order.

Fig. 4.1(a) shows the planar graph for the formula $E_1 \wedge E_2 \wedge E_3 \wedge E_4$, where $E_1 = (u_1 \vee \overline{u_2} \vee u_3)$, $E_2 = (\overline{u_3} \vee \overline{u_4} \vee u_5)$, $E_3 = (u_2 \vee u_3 \vee u_4)$, and $E_4 = (\overline{u_1} \vee u_4 \vee \overline{u_5})$ (and also for $2^{12} - 1$ other formulas that have the same variables in the same clauses and differ only in whether or not a particular occurrence is negated). Our proof requires an **orthogonal** embedding of the formula graph as shown in Fig. 4.1(b). Such an embedding can be constructed in linear time and with $O(n^2)$ total area [20] (see also [9]). Even though the orthogonal drawing assumes a graph whose maximum degree is four, a simple transformation can ensure this requirement without affecting the remainder of the argument: clause vertices have degree three by design and edges from a variable vertex to $k > 4$ clauses can be replaced by a tree with k leaves and interior nodes of degree ≤ 4.

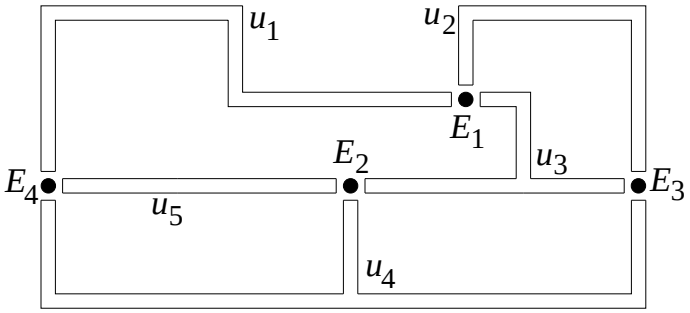
The demand points for the directional p-median instance are organized into n rectilinear polygons called **circuits**, one for each variable, and m additional points, one per clause. Fig. 4.1(c) illustrates the schematic based on the orthogonal embedding of Fig. 4.1(b) (analogous to the schematic used by Megiddo and Supowit [17]). Table 4.1 gives an overview of the elements of the reduction.



(a) The planar graph for an instance of planar 3-SAT.



(b) An orthogonal embedding of the above planar graph.



(c) The schematic for the p-median instance based on the planar 3-SAT instance.

Figure 4.1: Rectilinear embedding of the sample problem instance.

Planar 3-SAT	$\rightarrow$	DPM2
variable u_i	$\rightarrow$	circuit C^i of points $P^i_1, P^i_2, \ldots, P^i_{r_i}$
clause E_j	$\rightarrow$	clause point P^*_j
if u_i in E_j	$\rightarrow$	P^*_j is located near circuit C^i in a true configuration
if $\overline{u_i}$ in E_j	$\rightarrow$	P^*_j is located near circuit C^i in a false configuration

Table 4.1: Summary of the reduction from planar 3-SAT to DPM2.

4.2 Construction of the circuit for each variable

A 3-SAT variable u_i is represented by a circuit C^i of points $\{P^i_1, P^i_2, \ldots, P^i_{r_i}\}$, such that $r_i = 0 \pmod 6$ and $r_i \geq 18$. Circuits are designed in such a way that an optimal set of $r_i/3$ supply points partitions the demand points in one of two ways, to represent true and false values, respectively. In the circuit in Figure 4.2, subsets in the true (respectively, false) partition are enclosed with solid (respectively, dashed) lines.

Let $S \subseteq \{P^i_1, P^i_2, \ldots, P^i_{r_i}\}$ be any subset of a circuit C^i and let $f(S)$ denote the minimum of the directional 1-median problem on S (as in [17]). The optimal supply point for any subset S is the point $(x_{\max}, y_{\max})$, where $x_{\max} = \max_{P^i_k \in S}\{x_k\}$ and $y_{\max} = \max_{P^i_k \in S}\{y_k\}$, and $f(S)$ is the sum of the distances of all points in S to $(x_{\max}, y_{\max})$. In the **true partition,** every subset S is made up of three points such that $f(S) = 8$. The **false partition** has three-point subsets with f-value 8 along the straight segments, and subsets with two and four points, respectively, in the SW and NE corners. The latter have supply points that are not demand points (they are indicated by $\times$ in Figure 4.2) and have f-values of 4 and 12, respectively.

Although the circuit in Figure 4.2 has only four corners, this will not always be the case. For example, the circuit for u_3 in Figure 4.1(c) has ten corners: two NW, three NE, three SW, and two SE.

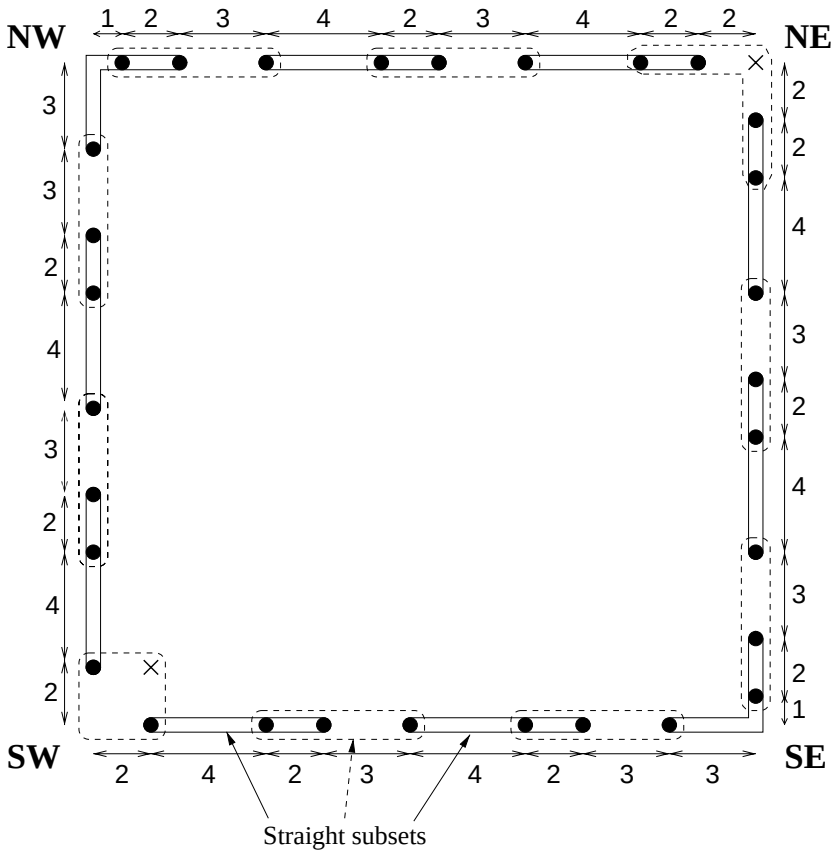


Figure 4.2: A simple circuit with four corners, one of each type. Solid lines indicate a true partition; the dashed lines indicate a false partition. The two ×'s mark supply points that do not coincide with demand points.

For technical reasons having to do with details of the proof, we construct circuits according to the following rules:

(1) If a circuit has r_i points then $r_i = 0 \pmod 6$ and $r_i \geq 18$.

(2) Points in corners of a circuit are arranged as in Figure 4.2, depending on the direction the corner faces (one of NW, NE, SW, or SE).

(3) Points are situated according to the following interpoint spacing rules:

   (a) The first point up or right from a SW corner is at distance 2; subsequent interpoint distances are 4,2,3,4,2,3, *etc.*, until a point at distance 3 from a NW or SE corner is reached.

   (b) The first point down or left from a NE corner is at distance 2; subsequent interpoint distances are 2,4,3,2,4,3, *etc.*, until a point at distance 1 from a NW or SE corner is reached.

(4) A side of a circuit must include at least one straight subset of either the true or the false partition.

(5) Sides of a circuit must be more than 6 units apart.

It is not hard to see that these conditions can be met with at most a constant multiplicative increase in the overall area of the graph embedding.

4.3 Alignment of clause points with variable circuits

For each clause $E = \ell_1 \vee \ell_2 \vee \ell_3$, we add a demand point P^* located where the circuits corresponding to the variables of ℓ_1, ℓ_2, and ℓ_3 come together. If $\ell_j = u$, for some variable u, then point P^* is positioned in one of the **true**

clause configurations (T_1, T_2, or T_3) with u's circuit C, shown in Figure 4.3. If $\ell_j = \overline{u}$, then point P^* is positioned in one of the **false** clause configurations (F_1, F_2, or F_3) with C, as Figure 4.4 illustrates. Figure 4.5 represents clause E_1 from the example given in Figure 4.1: clause point P_1^* is aligned with the u_1-circuit in configuration T_1, with the u_2-circuit in configuration F_3, and with the u_3-circuit in configuration T_2.

Similar to the proof in [17], we consider the directional p-median problem on some circuit C^i with $p = r_i/3$. We may rephrase the problem as: Partition C^i into p sets $S_1, S_2, \ldots, S_p$ so as to minimize $\sum f(S_j)$. For convenience, we call $f(S)$ the f-value of the subset, and we call $\sum f(S_j)$ the f-sum for the partition. For the true partition, the f-sum is equal to $8r_i/3$, because the f-value of each subset is 8. For the false partition, the f-sum is also equal to $8r_i/3$, because *on average* the f-value of each subset is 8. This result relies on the proof in Section 4.5.2 that there is an equal number of SW and NE corners in any circuit. Recall that, in the false partition, the two-point subset in the SW corner has f-value 4, and the four-point subset in the NE corner has f-value 12; thus each subset in the false partition *on average* contains three points and has f-value 8.

Each true (respectively, false) configuration is carefully constructed so that the point P^* can join a subset of the true (respectively, false) partition and add 12 to the f-value for that subset. Joining any other subset would result in a higher (non-optimal) f-sum for the circuit. Therefore, if the truth values are chosen so that E is satisfied, then there is at least one subset that P^* may join such that the circuit will have a new f-sum that is 12 more than its previous f-sum.

Suppose we have constructed a DPM2 instance from a 3-SAT instance

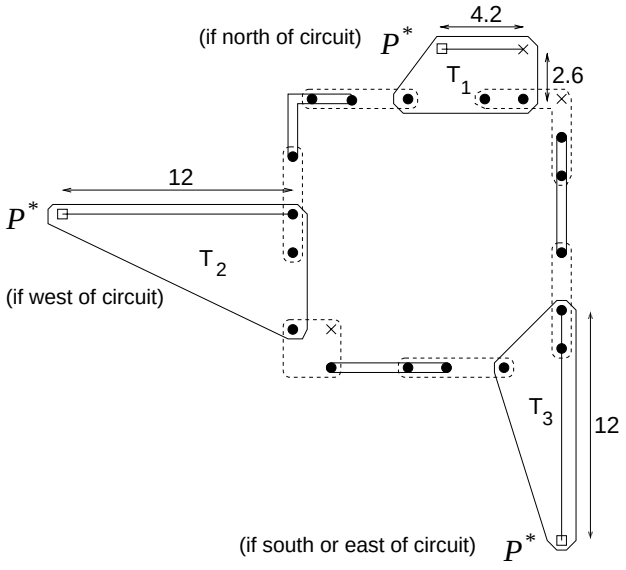


Figure 4.3: Possible locations of clause points for the true partition.

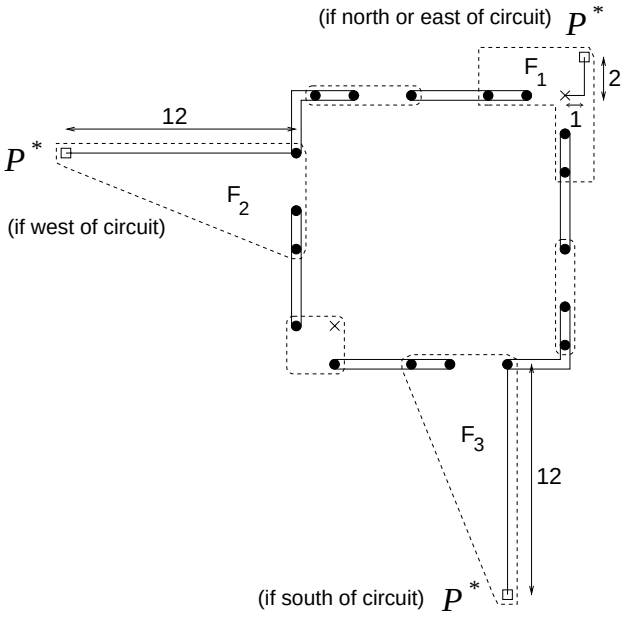


Figure 4.4: Possible locations of clause points for the false partition.

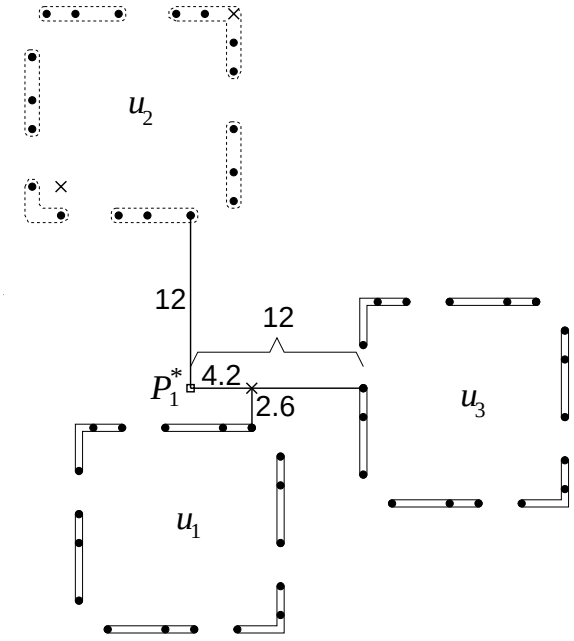


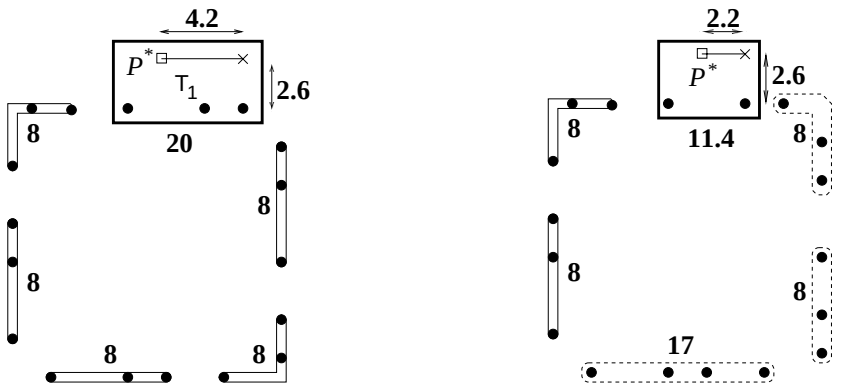
Figure 4.5: Position of clause point P_1^* for clause $E_1 = (u_1 \vee \overline{u_2} \vee u_3)$, from the example given in Figure 4.1.

with r total demand points in the variable circuits and m additional points for m 3-SAT clauses. The decision problem will ask whether or not there exist $p = r/3$ supply points with total cost (*i.e.* sum of f-sums across all circuits) at most $8r/3 + 12m$.

4.4 Sketch of proof

It is not hard to see that a satisfiable formula implies the existence of p supply points with total cost $8r/3 + 12m$. First, cluster the points in each variable circuit using either the true partition or the false partition, depending on whether the satisfying assignment makes that variable true or false. With no clause points, this leads to a solution with $r/3$ supply points and a cost of $8r/3$. Second, because each clause has at least one true literal, each clause point has a circuit with which it aligns so that it can join a cluster and add no more than 12 to the total cost. The total cost will therefore be $8r/3 + 12m$ as desired.

Now we need to ensure that any choice of $r/3$ supply points having cost $\leq 8r/3 + 12m$ corresponds to a satisfying solution for the 3-SAT formula. This will be argued in two stages. First, we observe that any partition of the demand points can only include **legitimate** clusters, ones that are part of either a true partition, a false partition, a T cluster (see Figure 4.3) on a circuit with a true partition, or an F cluster (see Figure 4.4) on a circuit with a false partition. Then the satisfying assignment can be derived from the true and false partitioning of the circuits for the variables; that it is a satisfying assignment follows from the fact that each clause point can only be part of a cluster belonging to a circuit with the "right" partition.



(a) Optimal (b) Non-optimal

Figure 4.6: Overall cost increases when an illegitimate cluster is chosen.

The first stage of this second proof direction requires careful analysis of many cases (full details are given in Section 4.5). The reader should become convinced that any cluster that is not legitimate will incur a penalty in f-value. This assertion is obvious for clusters that have legitimate ones as proper subsets (adding a new point to a cluster, unless it is the supply point, will increase its f-value). Less obvious is the analysis of clusters that are proper subsets of legitimate ones or that overlap with legitimate ones. For these, we argue that a local decrease, if any, is necessarily "paid for" by a larger increase in another part of the circuit. For example, Figure 4.6(b) shows an illegitimate cluster with local decrease in f-value involving the clause point P^* from the T_1 configuration, and demonstrates how this forces a larger increase elsewhere. The optimal partition for this circuit and clause point, as shown in Figure 4.6(a), has an f-sum of 60. The illegitimate cluster in 4.6(b) that includes the clause point has f-value 11.4 instead of 20, forcing another cluster to have f-value 17 instead of 8, making the total f-sum 60.4.

4.5 Proof details

4.5.1 Notation

To facilitate the proofs in the following sections, we introduce some notation that allows us to describe circuits and clause configurations more precisely.

Labeling the SW and NE corners. We wish to label the SW and NE corners in a circuit, while ignoring the NW and SE corners. Without loss of generality, we designate any SW corner of a circuit as the **beginning** corner of the circuit and label it L_1. We next traverse the circuit, starting by moving upwards away from L_1, and label the SW and NE corners as we come to them, with $L_2, L_3,$ *etc.*

Walking around a circuit. We wish to describe a walk around all or part of a circuit. Let $o_i = (\langle corner \rangle, \langle direction \rangle)$ denote an orientation on a circuit, where $\langle corner \rangle$ represents the type of corner where we are currently standing (NE, NW, SE, or SW), and $\langle direction \rangle$ represents the direction we are facing immediately after turning the corner (u, d, ℓ, or r, meaning up, down, left, or right). We may describe a (partial or complete) walk around a circuit by a sequence of orientations $o_1, o_2, \ldots, o_k$ for some integer k. For example, a clockwise walk around part of a square circuit, beginning at the SW corner (L_1) and ending at the SE corner, can be described as the sequence (SW,u),(NW,r),(NE,d),(SE,ℓ).

4.5.2 An equal number of SW and NE corners

We next consider the order in which NE and SW corners occur in a circuit.

from:\to:	SW,u	SW,r	NW,r	NW,d	NE,d	NE,ℓ	SE,ℓ	SE,u
SW,u			$\surd$			$\surd$		
SW,r					$\surd$			$\surd$
NW,r					$\surd$			$\surd$
NW,d		$\surd$					$\surd$	
NE,d		$\surd$					$\surd$	
NE,ℓ	$\surd$			$\surd$				
SE,ℓ	$\surd$			$\surd$				
SE,u			$\surd$			$\surd$		

Table 4.2: Possible moves in a walk around a circuit.

Lemma 4.1 $\{L_1, L_3, L_5, \ldots\}$ are SW corners, and $\{L_2, L_4, L_6, \ldots\}$ are NE corners.

Proof. We describe a walk around a circuit and exhaustively list the possibilities to show that the lemma is true. From a given current orientation, the possibilities for the next orientation are limited to those listed in Table 4.2.

We now examine four cases that prove the lemma.

Case (a): (SW,u). We cannot reach another SW corner without first reaching a NE corner. Using the table to list the possible next moves, we have:

(1) (NE,ℓ). Done.

(2) (NW,r),(NE,d). Done.

(3) [(NW,r),(SE,u)]^k,(NE,ℓ), for some integer $k \geq 1$. That is, we pass through the orientations [(NW,r),(SE,u)] a total of k times before arriving at (NE,ℓ). The fact that the circuit is a closed loop guarantees that k is finite, and thus ensures that we must eventually arrive at (NE,ℓ).

Case (b): (SW,r). We cannot reach another SW corner without first reaching a NE corner. Using the table to list the possible next moves, we have:

(1) (NE,d). Done.

(2) (SE,u),(NE,ℓ). Done.

(3) [(SE,u),(NW,r)]^k,(NE,d), for some integer $k \geq 1$. Done.

Case (c): (NE,d). We cannot reach another NE corner without first reaching a SW corner. Using the table to list the possible next moves, we have:

(1) (SW,r). Done.

(2) (SE,ℓ),(SW,u). Done.

(3) [(SE,ℓ),(NW,d)]^k,(SW,r), for some integer $k \geq 1$. Done.

Case (d): (NE,ℓ). We cannot reach another NE corner without first reaching a SW corner. Using the table to list the possible next moves, we have:

(1) (SW,u). Done.

(2) (NW,d),(SW,r). Done.

(3) [(NW,d),(SE,ℓ)]^k,(SW,u), for some integer $k \geq 1$. Done.

■

4.5.3 Notation revisited: A point numbering scheme

Lemma 4.1 implies that a circuit contains an equal number of NE corners and SW corners. Suppose a circuit has c SW corners (and therefore also c NE corners); then the last SW corner before reaching the beginning of the circuit is L_{2c-1}, and the last NE corner is L_{2c}. The numbering is $(\mathrm{mod}\, 2c)$, such that $L_{2c+1} = L_1$. The SW and NE corners break the circuit into alternating segments: for $k = 1, 2, 3, \ldots, c$, the segment from L_{2k-1} to L_{2k} has interpoint distances in the (repeating) sequence [4,2,3], whereas the segment from L_{2k} to L_{2k+1} has interpoint distances in the (repeating) sequence [2,4,3].

Now we wish to label the points in a circuit in such a way that, for each integer $k = 1 \pmod 3$, we have $d_{dr}(P_k, P_{k+1}) = 4$, $d_{dr}(P_{k+1}, P_{k+2}) = 2$, and $d_{dr}(P_{k+2}, P_{k+3}) = 3$. We define $n[s, s+1]$ to be the number of points between corners L_s and L_{s+1}, $s = 1, 2, \ldots, 2c$. As a result of the way circuits are constructed, $n[s, s+1] = 0 \pmod 3$ for all s.

Starting at L_1, we label points in increasing order on the segments from L_{2s-1} to L_{2s}, where $s = 1, 2, \ldots, c$. From L_1 to L_2, points are labeled $P_1, P_2, \ldots, P_{n[1,2]}$. We next label the points between L_3 and L_4 as follows: $P_{n[1,2]+1}, P_{n[1,2]+2}, \ldots, P_{n[1,2]+n[3,4]}$. After labeling the points on the segment from L_{2c-1} to L_{2c}, the last point of which will be P_g, $g = \sum_{s=1}^{c} n[2s-1, 2s]$, we next start at L_1 and traverse the circuit in the opposite direction, labeling the points on the segments we skipped. Thus we start with P_{g+1} and label points in increasing order on the segments from L_{2s+1} to L_{2s}, where $s = c, c-1, \ldots, 1$. The last point labeled will be P_{r_i}, $r_i = \sum_{s=1}^{2c} n[s, s+1]$. Figure 4.7 shows a circuit with ten SW/NE corners and 66 points numbered according to this scheme.

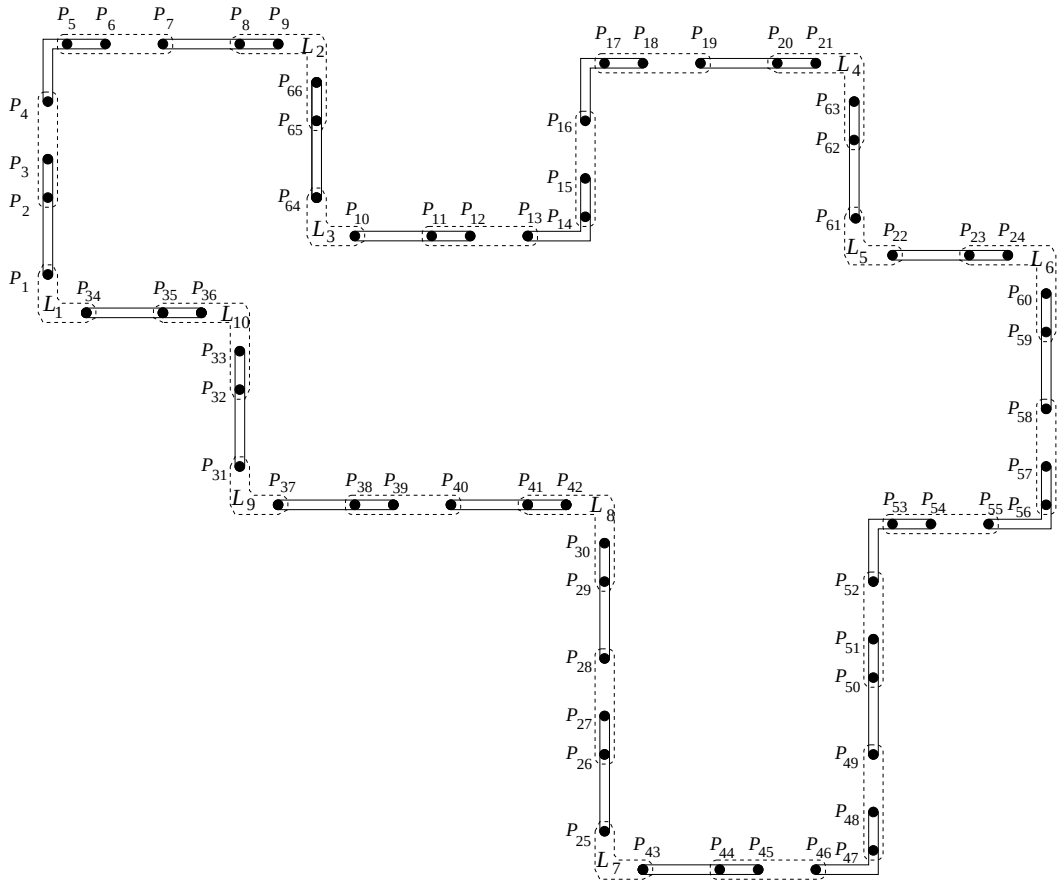


Figure 4.7: An example of the point numbering scheme.

4.5.4 A circuit lemma: The true partition and the false partition are the only optimal partitions

We need only consider subsets of adjacent points in a circuit; we call such subsets **adjacent subsets**. A **non-adjacent subset** is a subset whose convex hull encloses one or more points not in the subset. Clearly, such a subset is never optimal, since one or more of the points has been assigned to a supply point that is not its closest supply point.

Lemma 4.2 The true partition and the false partition are the only partitions that yield the minimum f-sum of $8r_i/3$.

Proof. We examine all adjacent subsets, legitimate and illegitimate.

Case (a): A subset of three points that includes both SW corner points has f-value of at least 14. An adjacent subset is formed by letting the third point be one of the points adjacent to a SW corner point, located four units away from the nearest corner point. The f-value of such a subset is exactly 14.

Case (b): A subset of three points that includes at least two NE corner points has f-value of at least 8. For an adjacent subset, the third point must be either another NE corner point (in which case the f-value is 8) or a point P_k, where $k = 1 \pmod 3$, located four units away from the nearest corner point (in which case the f-value is again 8).

Case (c): A subset of three points (P_k, P_{k+1}, P_{k+2}) that includes less than two SW corner points and includes less than two NE corner points and $k \neq 0 (\mathrm{mod} 3)$, has f-value of 8. If $k = 1 (\mathrm{mod} 3)$, then $d_{dr}(P_k, P_{k+1}) = 4$

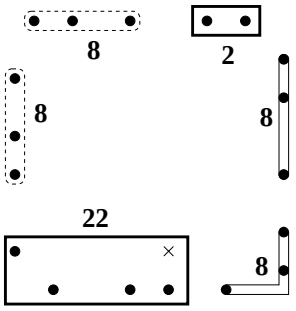


Figure 4.8: Lemma 4.2(e): A non-optimal subset of four points that contains both SW corner points has f-value of 22.

and $d_{dr}(P_{k+1}, P_{k+2}) = 2$. P_{k+2} serves as the solution point and the subset has f-value of 8. If $k = 2 \pmod 3$, then $d_{dr}(P_k, P_{k+1}) = 2$ and $d_{dr}(P_{k+1}, P_{k+2}) = 3$. P_{k+2} serves as the solution point and the subset has f-value of 8.

Case (d): A subset of three points (P_k, P_{k+1}, P_{k+2}) that includes less than two SW corner points and includes less than two NE corner points and $k = 0 \pmod 3$, has f-value of 11. P_{k+2} serves as the solution point. Since $d_{dr}(P_k, P_{k+1}) = 3$ and $d_{dr}(P_{k+1}, P_{k+2}) = 4$, the f-value of the subset is 11.

Case (e): A subset of four points that contains both SW corner points has f-value of at least 22. Figure 4.8 shows this subset as a part of a partition on a circuit of 18 points into 6 subsets. Any partition containing this subset can do no better than an f-sum of 56. In contrast, an optimal partition (the true or false partition) achieves an f-sum of 48.

Case (f): A subset of four points that contains all four NE corner points has f-value of 12.

Case (g): A subset of four points that contains exactly three NE corner points has f-value of 16.

Case (h): A subset of four points that contains less than three NE corner points and less than two SW corner points has f-value of at least 17. Let $(P_k, P_{k+1}, P_{k+2}, P_{k+3})$ be a subset of size four. If $k = 2 \pmod 3$ then the f-value is 20. If $k = 1 \pmod 3$ or $k = 0 \pmod 3$, the f-value of the subset is 17.

Case (i): A subset of two points that is comprised of both SW corner points has f-value of 4. ■

4.5.5 Two clause configuration lemmas: Only legitimate subsets may be in an optimal partition

Lemma 4.3 Adding a clause point P_j^* to a legitimate subset in one of the configurations shown in Figures 4.3 and 4.4 (*i.e.*, F_1, F_2, F_3, T_1, T_2, T_3) results in an f-sum increase of 12 for that subset.

Proof. We examine each case.

Case (a): A subset of five points that includes the four NE corner points and a clause point P_j^* located in configuration F_1 has f-value of 24. Before the addition of P_j^*, the NE corner subset has f-value of 12. P_j^*, located three units northeast of the subset's former solution point (marked by $\times$), becomes the new supply point.

Case (b): A subset of four points that includes the three points (P_k, P_{k+1}, P_{k+2}), $k = 2 \pmod 3$, and a clause point P_j^* located in configuration F_2 or F_3

has f-value of 20. The point P_{k+2} serves as the supply point.

Case (c): A subset of four points that includes the three points (P_k, P_{k+1}, P_{k+2}), $k = 1 \pmod 3$, and a clause point P_j^* located in configuration T_1 has f-value of 20. The $\times$ shown in Figure 4.3 serves as the supply point. P_j^* contributes 4.2 to the f-value, while points P_k, P_{k+1}, and P_{k+2} contribute 8.6, 4.6, and 2.6, respectively.

Case (d): A subset of four points that includes the three points (P_k, P_{k+1}, P_{k+2}), $k = 1 \pmod 3$, and a clause point P_j^* located in configuration T_2 or T_3 has f-value of 20. The point P_{k+2} serves as the supply point. ■

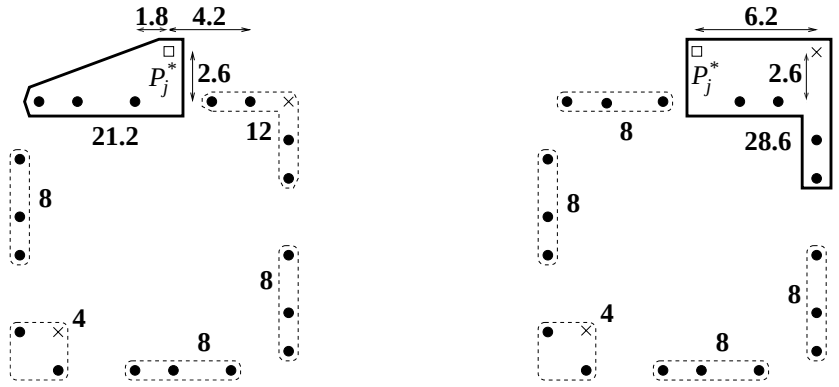
Lemma 4.4 Combining the clause point P_j^* with any subset other than those shown in Figures 4.3 and 4.4 will result in a partition that fails to reach the minimum f-sum.

Proof. We examine each case.

Case (a): A subset of four points that includes the three points (P_k, P_{k+1}, P_{k+2}), $k = 2 (\mathrm{mod} 3)$, and a clause point P_j^* located in configuration T_1 (shown in Figure 4.9(a)) has f-value of 21.2. The point P_j^* serves as the supply point.

Case (b): A subset of five points that includes the four NE corner points and a clause point P_j^* located in configuration T_1 (shown in Figure 4.9(b)) has f-value of 28.6. The $\times$ serves as the supply point.

Case (c): A subset of four points that includes the three points (P_k, P_{k+1}, P_{k+2}), $k = 2 (\mathrm{mod} 3)$, and a clause point P_j^* located in configuration T_2 (shown



(a) Lemma 4.4(a)

(b) Lemma 4.4(b)

Figure 4.9: Two non-optimal subsets involving clause point P_j^* located in configuration T_1.

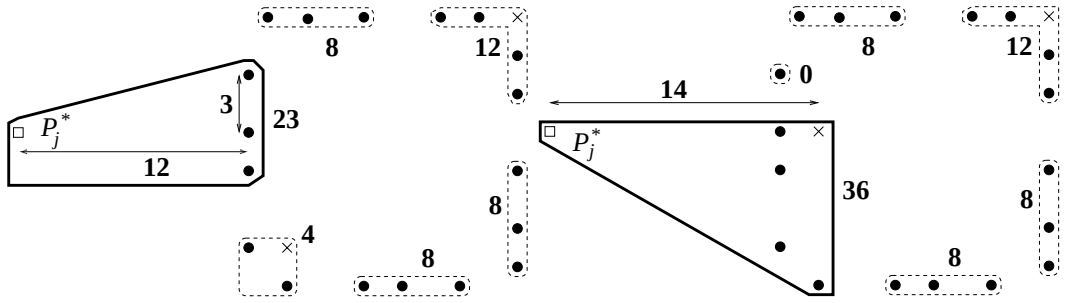
in Figure 4.10(a)) has f-value of 23. The point P_{k+2} serves as the supply point.

Case (d): A subset of five points that includes both SW corner points, the two points (P_k, P_{k+1}), $k = 2 \pmod 3$, and a clause point P_j^* located in configuration T_2 (shown in Figure 4.10(b)) has f-value of 36. The $\times$ serves as the supply point.

Case (e): A subset of four points that includes the three points (P_k, P_{k+1}, P_{k+2}), $k = 2 (\mathrm{mod} 3)$, and a clause point P_j^* located in configuration T_3 (shown in Figure 4.11(a)) has f-value of 26. The $\times$ serves as the supply point.

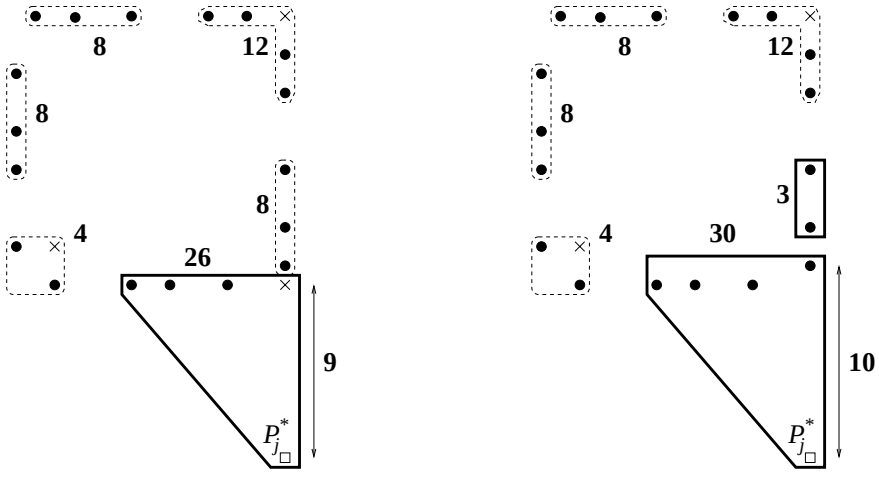
Case (f): A subset of five points that includes the four points $(P_k, P_{k+1}, P_{k+2}, P_{k+3})$, $k = 2 (\mathrm{mod} 3)$, and a clause point P_j^* located in configuration T_3 (shown in Figure 4.11(b)) has f-value of 30. The $\times$ serves as the supply point.

Case (g): A subset of four points that includes the three points (P_k, P_{k+1}, P_{k+2}),



(a) Lemma 4.4(c) (b) Lemma 4.4(d)

Figure 4.10: Two non-optimal subsets involving clause point P_j^* located in configuration T_2.



(a) Lemma 4.4(e) (b) Lemma 4.4(f)

Figure 4.11: Two non-optimal subsets involving clause point P_j^* located in configuration T_3.

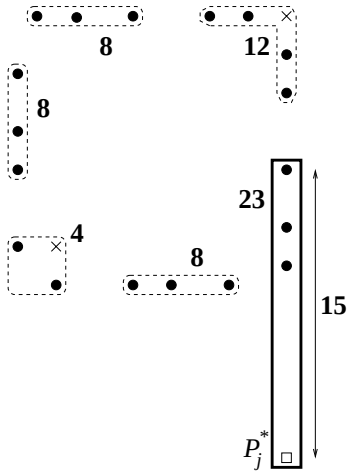


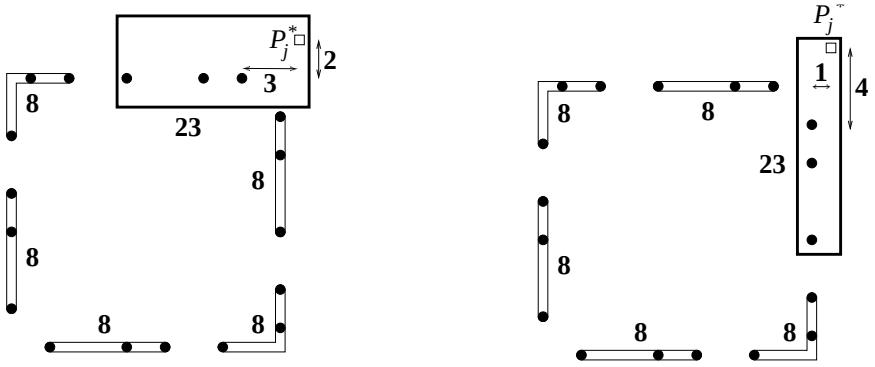
Figure 4.12: Lemma 4.4(g): A third non-optimal subset involving clause point P_j^* located in configuration T_3.

$k = 2(\text{mod}3)$, and a clause point P_j^* located in configuration T_3 (shown in Figure 4.12) has f-value of 23. The point P_{k+2} serves as the supply point.

Case (h): A subset of four points that includes the three points (P_k, P_{k+1}, P_{k+2}), $k = 1(\text{mod}3)$, and a clause point P_j^* located in configuration F_1 (shown in Figure 4.13(a)) has f-value of 23. The point P_j^* serves as the supply point.

Case (i): A subset of four points that includes the three points (P_k, P_{k+1}, P_{k+2}), $k = 1(\text{mod}3)$, and a clause point P_j^* located in configuration F_1 (shown in Figure 4.13(b)) has f-value of 23. The point P_j^* serves as the supply point.

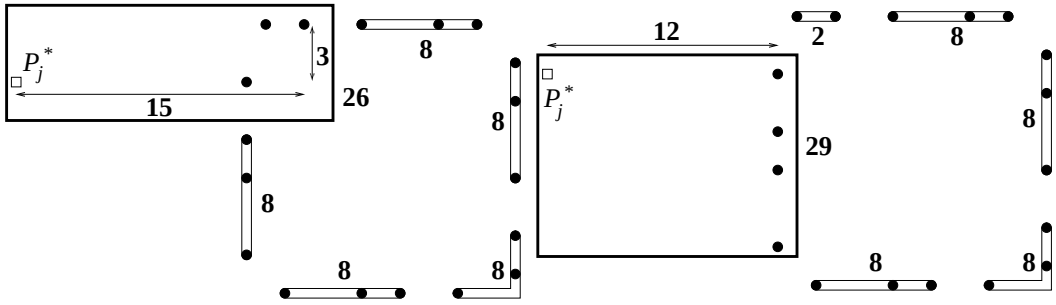
Case (j): A subset of four points that includes the three points (P_k, P_{k+1}, P_{k+2}), $k = 1(\text{mod}3)$, and a clause point P_j^* located in configuration F_2 (shown in Figure 4.14(a)) has f-value of 26. The point P_{k+2} is the supply point.



(a) Lemma 4.4(h)

(b) Lemma 4.4(i)

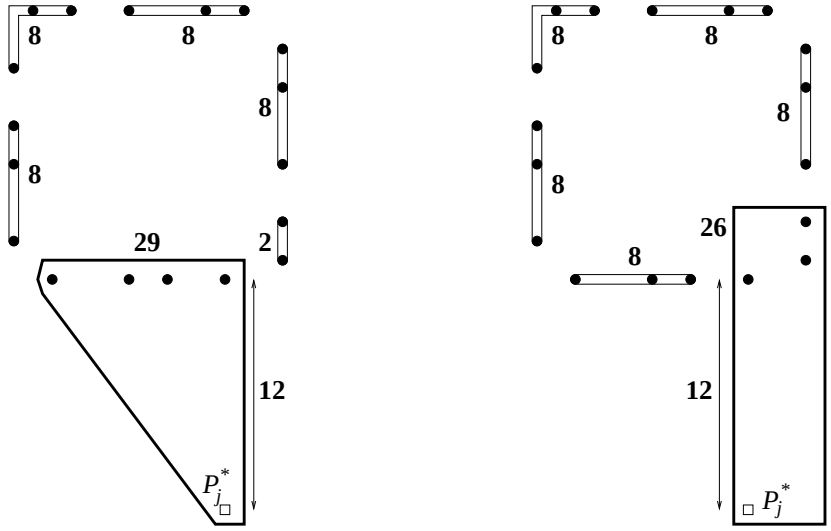
Figure 4.13: Two non-optimal subsets involving clause point P_j^* located in configuration F_1.



(a) Lemma 4.4(j)

(b) Lemma 4.4(k)

Figure 4.14: Two non-optimal subsets involving clause point P_j^* located in configuration F_2.



(a) Lemma 4.4(ℓ) (b) Lemma 4.4(m)

Figure 4.15: Two non-optimal subsets involving clause point P_j^* located in configuration F_3.

Case (k): A subset of five points that includes the four points $(P_k, P_{k+1}, P_{k+2}, P_{k+3})$, $k = 1 (\text{mod} 3)$, and a clause point P_j^* located in configuration F_2 (shown in Figure 4.14(b)) has f-value of 29. The point P_{k+3} is the supply point.

Case (ℓ): A subset of five points that includes the four points $(P_k, P_{k+1}, P_{k+2}, P_{k+3})$, $k = 1 (\text{mod} 3)$, and a clause point P_j^* located in configuration F_3 (shown in Figure 4.15(a)) has f-value of 29. The point P_{k+3} is the supply point.

Case (m): A subset of four points that includes the three points (P_k, P_{k+1}, P_{k+2}), $k = 1 (\text{mod} 3)$, and a clause point P_j^* located in configuration F_3 (shown in Figure 4.15(b)) has f-value of 26. The point P_{k+2} is the supply point.

■

Chapter 5

An efficient heuristic algorithm for DPM2

In Chapter 4, we showed the directional rectilinear p-median problem in two dimensions (DPM2) to be NP-complete. In this chapter we present an efficient heuristic algorithm that produces good quality results for a variety of input distributions. The new heuristic, called Teitz & Bart Restricted Heuristic (TBr), makes use of the vertex substitution heuristic from Teitz & Bart (TB) [26], described in Section 5.2, and the heuristic concentration approach (HC) from Rosing & ReVelle [24], described in Section 5.3. Finally, in Section 5.5 we present the results of a simulation study that evaluates our new heuristic under a variety of input conditions.

5.1 Effect of distance characteristics on computational effort

Recall that, for a discrete p-median problem (Discrete-PM2, defined in Section 1.2), the c points eligible to serve as supply points are called **can-**

didate points. The $n \times c$ distance matrix $[d_{ij}]$ holds the distances from each demand point to each candidate point. Different distance metrics produce distance matrices with different characteristics, upon which the performance of a heuristic depends. In [25], the authors identify two such characteristics: symmetry and the ability to satisfy the triangle inequality. The lack of symmetry in the distance matrix has a minor impact on performance, but failure to satisfy the triangle inequality is a much graver crime, making optimal or even good quality solutions hard to find. (Non-directional) rectilinear and Euclidean distance matrices both are symmetric and obey the triangle inequality. A randomly generated distance matrix in general will neither be symmetric nor obey the triangle inequality. As for our directional rectilinear distance metric, the outlook is positive as regards the most critical attribute: the distance matrix obeys the triangle inequality but fails to be symmetric (recall that if $d_{dr}((x_i, y_i), (x_j, y_j))$ is finite and > 0, then $d_{dr}((x_j, y_j), (x_i, y_i))$ is infinite).

ReVelle labels certain Discrete-PM2 problems as **integer friendly**, meaning that "either integer termination of linear programming formulations are frequent, or little branch and bound is needed to resolve the problem in integers" [21]. The conclusions of [25] are that the characteristic of obeying the triangle inequality has greatest impact on the integer friendliness of an instance of Discrete-PM2. The question of why this is the case remains open.

5.2 Teitz & Bart vertex substitution heuristic for p-median

The 1968 Teitz & Bart [26] vertex substitution heuristic (TB) for Discrete-PM2 is well-known and much studied. As is the case with any heuristic, TB may become trapped in a local minimum. However, such pitfalls are

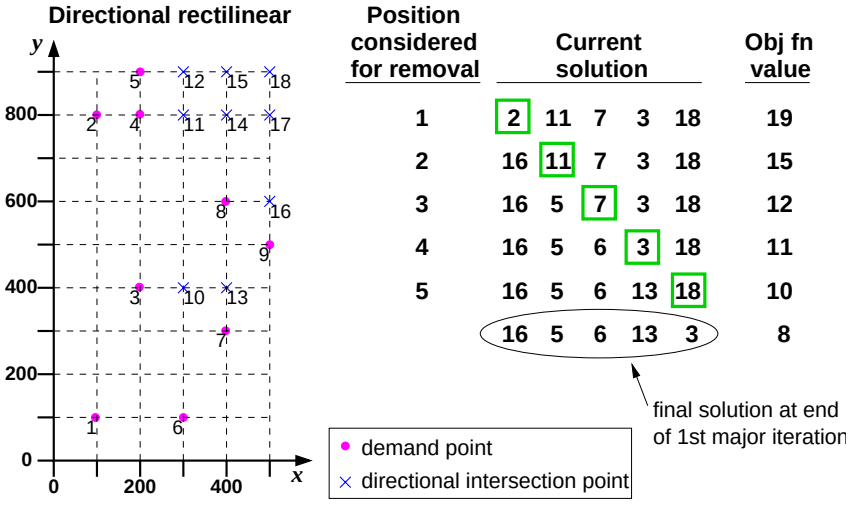


Figure 5.1: One major iteration of the TB heuristic on a sample problem.

rare, according to a study comparing TB to exact methods [23].

Figure 5.1 illustrates the operation of the TB heuristic on a sample DPM2 problem instance of $n = 9$ and $p = 5$.[1] TB begins with an initial solution of p supply points, chosen from among all candidate points. The heuristic assigns each demand point to its nearest supply point and evaluates the objective function for this assignment. The supply points are ordered arbitrarily. Beginning with the supply point in the first position, the heuristic tries substituting all other possible candidate points into this position; it replaces the first supply point with the point that causes the greatest decrease in the objective function. The heuristic then repeats this process with the supply point in the second position, then the third position, and so on. At each step, the heuristic tries to find the best supply point for the current position, given that all other supply points in the solution are fixed. After each of the p points in the solution have been considered for removal, one major iteration of the heuristic is

[1] Although TB was created for the non-directional p-median problem, it works similarly for the directional problem.

complete. The heuristic then begins the second major iteration, by considering the supply point in the first position for removal. The TB heuristic terminates when an iteration results in no changes to the solution, usually within only a few (≤ 5) iterations.

If $c = |C|$ is the number of candidate points, then each major iteration of TB runs in time $O(np(c-p))$, or $O(n^2p)$ if the set of demand points is also the set of candidate points [5]. There are p existing supply points and $c-p$ alternate candidate points, for a total of $p(c-p)$ pairs to be considered. For each of these pairs, we examine each demand point in order to find its closest supply point.

The performance of TB depends on the starting solution. A common approach is to generate a number of random starting solutions as input for multiple TB runs, and then to choose the best solution from among the local optima that are found [10].

The Densham & Rushton Global/Regional Interchange Algorithm (GRIA) improves the runtime of TB by taking advantage of characteristics of five well-known PM heuristics [6]. Speedup is achieved through the use of novel data structures that enable the heuristic to evaluate far fewer supply point substitutions. GRIA iterates between a global and a regional phase until an iteration of both phases yields no change. For both the non-directional problem, Discrete-PM2, and the directional problem, DPM2, the global phase terminates with a feasible solution, *i.e.* one in which each demand point is assigned to the "nearest" supply point (according to the appropriate distance metric). Next, during the regional phase, each subset of points assigned to a supply point is considered as a separate 1-median problem; for Discrete-PM2, the solution to this 1-median problem could be different from the current supply

point, so making an exchange improves the overall objective function.

However in DPM2, during the regional phase, no exchange will ever take place, since there will never be more than one valid directional median for a given subset of demand points. The only feasible 1-median for a subset of points is the point with (x_{max}, y_{max}), where x_{max} (respectively, y_{max}) is the maximum-valued x (respectively, y) from among all points in the subset. Therefore GRIA is not appropriate for a directional p-median problem.

5.3 Heuristic concentration

Rosing & ReVelle present a new methodology called Heuristic Concentration (HC) [24]. Although they demonstrate how HC works by applying it to Discrete-PM2, HC, like other metaheuristics such as Tabu Search and Simulated Annealing, can be applied to many different combinatorial problems. HC attempts to glean information from the many local minima obtained from repeated runs of a heuristic. For example, the (local minima) solutions resulting from separate TB runs may have differing objective function values, as well as different supply points in the solution set. However, Rosing & ReVelle discover that there is frequently a great deal of overlap in the solution sets corresponding to these local minima. HC takes advantage of this fact by taking a two-stage approach. First, HC builds a Concentration Set (CS) by taking the union of the several local minima solutions. The CS has a high likelihood of containing the supply points that make up the optimal solution set. The second stage then locates the best solution to the new subproblem, selecting a solution only from among the members of the CS.

Rosing & ReVelle randomly generate coordinate pairs and use the

Euclidean distance measure, for problems of size $n = 100, 125, 150, \ldots, 300$ with $p = 5, 10, 15, \ldots, 50$. They solve each problem optimally as well as heuristically. For each problem, they create the CS from the best five solutions of 200 TB runs. In the second stage, they use an integer linear program to optimally select the best solution from the CS.

5.4 The Teitz & Bart Restricted heuristic for DPM2

The motivation for designing a new algorithm for DPM2 is to find a very fast heuristic that doesn't sacrifice much in solution quality. Recall that the c candidate points in DPM2 include all n demand points as well as all directional intersection points, as described in Section 1.3. In the worst case, there could be $(n^2+n)/2$ candidate points to be considered for inclusion in the solution set of p supply points.

The Teitz & Bart Restricted heuristic (TBr) follows the two-stage approach of heuristic concentration. In the first stage, the concentration set (CS) is built in the following manner. We run the 1-dimensional algorithm for DPM1 (from Section 3.4) twice, once on the x-values and once on the y-values of the n demand points, to obtain the p best x's and the p best y's. Crossing these two sets yields p^2 points that will serve as the CS. In the second stage, we randomly generate 100 initial solutions from among all possible candidate points, *i.e.* the set C (n demand points plus up to $(n^2 - n)/2$ directional intersection points). Each initial solution serves as input into a separate run of the TB heuristic, which only chooses points for exchange from the CS. The final TBr solution is the best outcome from the 100 TB runs. Note that the final TBr solution may include a point that is not in the CS: each initial solution is

drawn from *all* candidate points, and then TB looks only in the CS for possible replacement points.

Complexity analysis

Let n be the number of demand points, c the number of candidate points (demand points plus directional intersection points), and p the number of supply points in the solution. Recall that c is on the order of n^2. Each major iteration of TBr runs in time $O(p(p^2-p)n)$ or $O(np^3)$. In contrast, each major iteration of TB runs in time $O(p(c-p)n)$ or $O(n^3p)$.

Building the concentration set during the first stage of TBr requires time $O\left(n\sqrt{p\log n}\right)$, which is the time required by the algorithm in [1] to solve DPM1.

5.5 Evaluation of the Teitz & Bart Restricted heuristic

Simulation set-up and input parameters

To determine the penalty in terms of excess load resulting from quantization, we designed a simulation study using a variety of different types of demand sets X. To generate points in the plane, we chose one discrete probability density function (pdf) for the x-values and one for the y-values, from among the possible pdf's given in Table 5.1. Each discrete distribution can generate integers in the range [1,1000]. The Bimodal distribution has one peak on the interval [251,350] and another at [651,750]. The peaks of the Bimodal distribution contain 80% of the density (that is, the probability of a Bimodal random variate taking on one of the 200 values from $\{251,252,\ldots,350\}\cup\{651,652,\ldots,750\}$ is

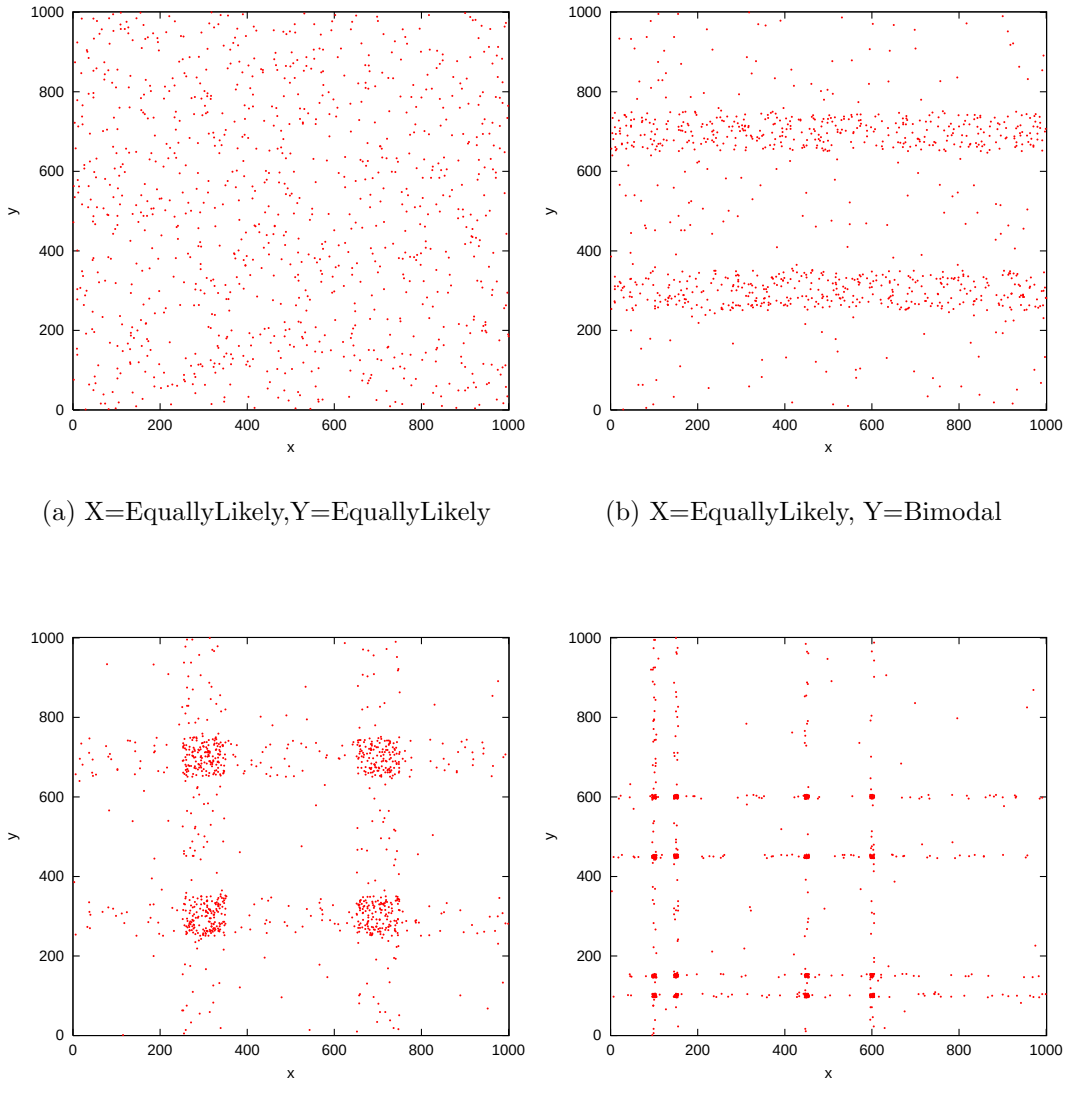
.8).[2] In contrast, the Quadrimodal distribution has four much narrower peaks, located on the intervals [96,105], [146,155], [446,455], and [596,605]. The peaks of the Quadrimodal distribution contain 80% of the density (that is, the probability of a Quadrimodal random variate taking on one of the 40 values from {96,97,. . .,105}∪{146,47,. . . ,155}∪{446,447,. . .,455}∪{596,597,. . .,605} is .8).

Four different combinations of the input distributions were selected to generate input sets X for the simulations. In the first (EE), both the x-values and the y-values were drawn from the EquallyLikely distribution. In the second (EB), the x-values are EquallyLikely and the y-values are Bimodal. In the third (BB), both the x-values and the y-values are Bimodal. In the fourth (QQ), both the x-values and the y-values are Quadrimodal. Figure 5.2 shows scatter plots of demand sets of size $n = 1000$ generated from the four input combinations used in the simulations.

From each input distribution, we generated fifty demand sets with $n = 100$ and another fifty demand sets with $n = 200$. Each demand set was generated starting from a unique seed for a Lehmer random number generator with modulus $2^{31} - 1$ and multiplier 48271.

Each demand set then served as input to the TB and TBr heuristics. In an attempt to obtain a solution as close to optimal as possible, the TB result for each demand set is actually the best of 100 runs of the TB heuristic. Analogous to the normalized quantization load for DPM1, defined in Equation (3.4), we define the **normalized ratio** as the measure of the performance penalty due to quantization for DPM2. The denominator of the normalized ratio is the density of X, $\rho_X = \sum_{i=1}^{n} x_i + \sum_{i=1}^{n} y_i$, which represents the amount of resources

[2]The discrete bimodal distribution is analogous to the continuous bimodal distribution used in Chapter 3.3, which has domain (0,1) and peaks over the intervals (.25,.35) and (.65,.75).



(a) X=EquallyLikely,Y=EquallyLikely

(b) X=EquallyLikely, Y=Bimodal

(c) X=Bimodal, Y=Bimodal

(d) X=Quadrimodal, Y=Quadrimodal

Figure 5.2: Scatter plots of $n = 1000$ for the four input distributions.

Distribution	$f(x)$	domain
EquallyLikely	1/1000	$[1, 1000]$
Bimodal	1/4000	$[1, 250]$, $[351, 650]$, $[751, 1000]$
	16/4000	$[251, 350]$, $[651, 750]$
Quadrimodal	5/24000	$[1, 95]$, $[106, 145]$, $[156, 445]$, $[456, 595]$, $[606, 1000]$
	480/24000	$[96, 105]$, $[146, 155]$, $[446, 455]$, $[596, 605]$

Table 5.1: Probability density function for input distributions.

needed to meet the demands of X *before* quantization. The numerator represents the amount of resources needed *after* quantization; it is equal to ρ_X plus the value of the DPM2 objective function given in Section 4.1. Letting n_j be the number of demand points mapped to supply point (z_j, t_j), $j = 1, \ldots, p$, we have:

$$\text{normalized ratio} = \frac{\rho_X + \sum_{i=1}^{n} \min_{1 \le j \le p} \{d_{dr}((x_i, y_i), (z_j, t_j))\}}{\rho_X} \quad (5.1)$$

$$= \frac{\sum_{j=1}^{p} n_j (z_j + t_j)}{\rho_X} \quad (5.2)$$

$$\geq 1 \quad (5.3)$$

The closer the normalized ratio is to one, the fewer resources are wasted. For each demand set, we calculated the normalized ratio for the values of $p = 5, 10, 15, 20, 25$. In addition, we calculated the normalized ratio for $p = 30$ for (1) all demand sets of size $n = 100$, and (2) QQ demand sets of size $n = 200$.

Simulation results

There are three sets of figures containing the results. Each set presents the complete information of the simulation runs, but each highlights a different feature of the data.

The first set contains two figures. Figure 5.3 (respectively, Figure 5.4) shows all the results for problem instances of size $n = 100$ (respectively, size $n = 200$). The normalized ratio is plotted against p for both the TB and the TBr heuristics, for each of the four input distribution combinations. Each point is the mean of 50 problem instances, and the error bars around each point designate a 95% confidence interval. We notice that:

(1) For all values of p, the curves for input EE are the highest, followed by EB and then BB.

(2) For values of $p > 15$, the curves for input QQ are the lowest.

We can attribute these features to the nature of the input. In the sample scatter plots for each input shown in Figure 5.2, the level of "order" increases as we move from EE to EB to BB to QQ. (Or, said another way, the level of randomness decreases as we move from EE to EB to BB to QQ.) The EE input appears to be the worst case scenario, with points scattered evenly over the plane. In contrast, the other inputs possess natural clusters where points fall with higher density: EB has two horizontal strips, BB has four squares, and QQ has 16 squares. Observation #2 above results from the fact that p must be at least 16 in order to take advantage of the 16 natural clusters within QQ.

In the next set of figures (Figures 5.5-5.8) we plot the same means (plus confidence intervals) as shown in Figures 5.3 and 5.4, but the plots are organized this time by input distribution combination. Figure 5.5 shows the all the results for the EE input distribution combination: normalized ratio vs. p for the TB and TBr heuristics, for problem instances of size $n = 100$ and $n = 200$. Similarly, Figures 5.6, 5.7, and 5.8 correspond to the EB, BB, and QQ inputs. We make two observations about these plots:

(1) For each input, the $n = 200$ curve lies above the $n = 100$ curve.

(2) The TBr curve closely tracks (lies slightly above) the TB curve.

Holding p fixed, it is reasonable to expect a higher quantization penalty for a demand set of size $n = 200$ as compared to $n = 100$. Observation #2 illustrates the quality of the TBr solution; we pay very little penalty in solution quality for using TBr instead of TB, and we realize great gains in speed.

The final set of plots includes Figures 5.9-5.16, which contain two graphs apiece. Each graph contains level curves in p of the normalized ratio for each of the 50 input instances (which, when averaged, yield the means given in previous plots). As we might expect, the variation diminishes from the $n = 100$ plot to the $n = 200$ plot.

.

5.6 Conclusion

In this chapter, we presented the Teitz & Bart Restricted (TBr) heuristic for the directional rectilinear p-median problem in two dimensions. Using as building blocks the 1968 Teitz & Bart (TB) heuristic [26] and the recent Heuristic Concentration metaheuristic from Rosing & ReVelle [24], we created TBr to be a faster alternative for applications with very large demand sets. Although TB consistently returned solutions of slightly better quality, TBr tracked TB's performance closely and enjoyed a significant speedup, running in time $O(np^3)$ as compared to TB's time of $O(n^3p)$.

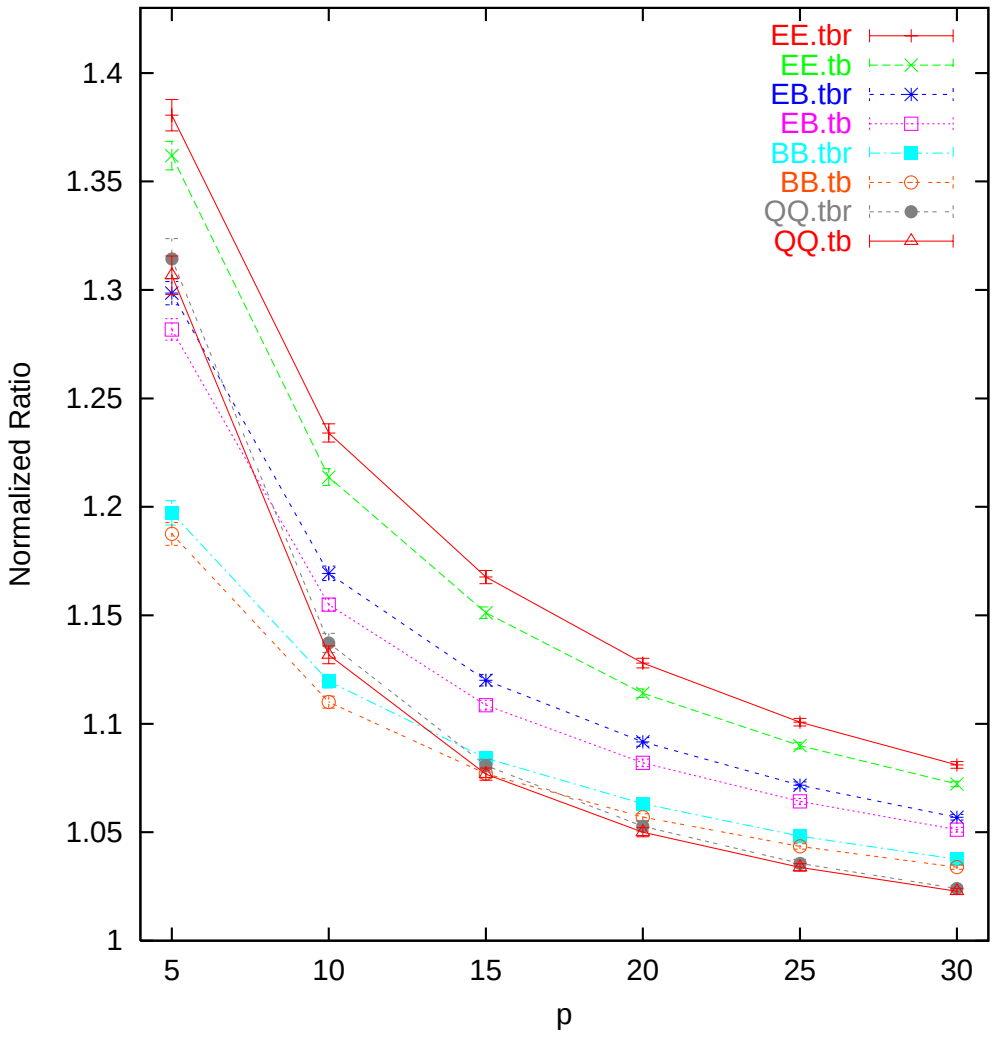


Figure 5.3: The normalized ratio vs. p, for the TB and TBr heuristics, for the four input distribution combinations. Each point (with 95% confidence interval) is the mean of 50 problem instances with $n = 100$.

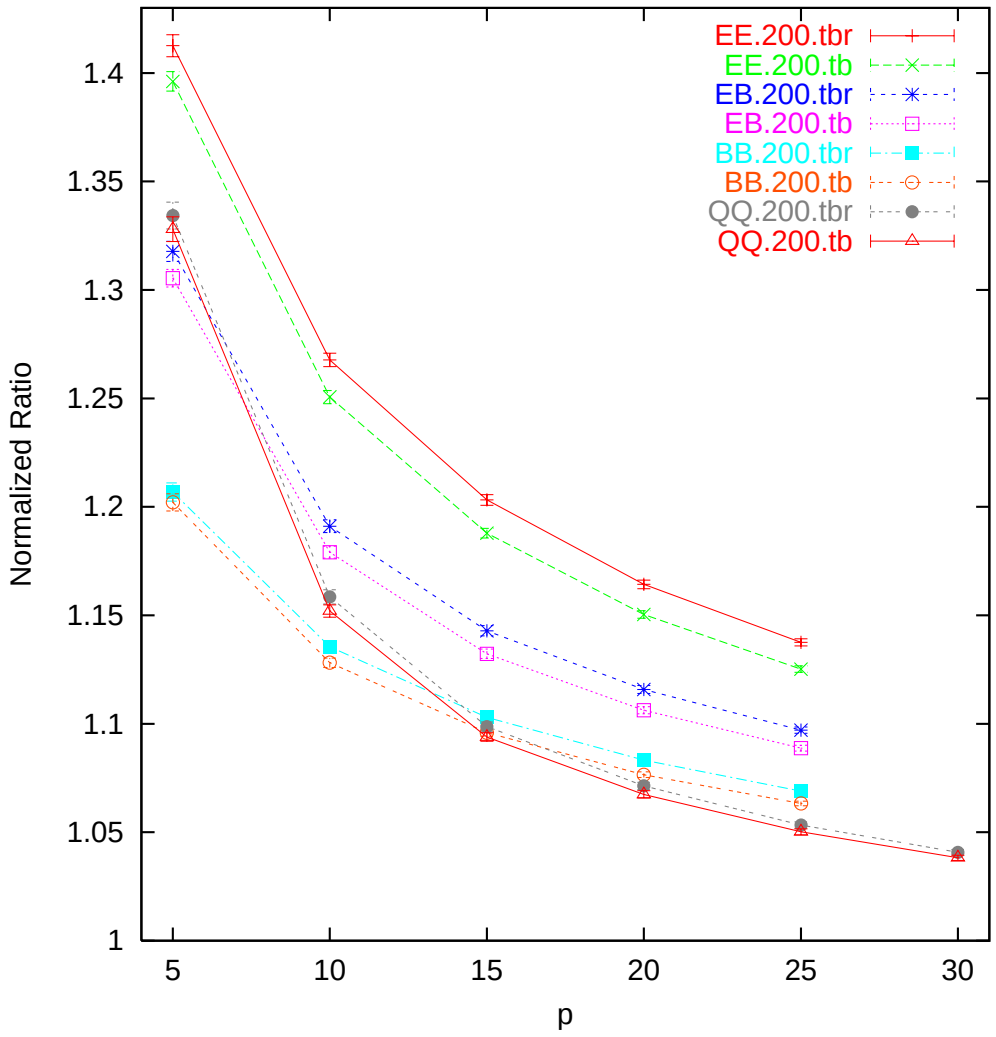


Figure 5.4: The normalized ratio vs. p, for the TB and TBr heuristics, for the four input distribution combinations. Each point (with 95% confidence interval) is the mean of 50 problem instances with $n = 200$.

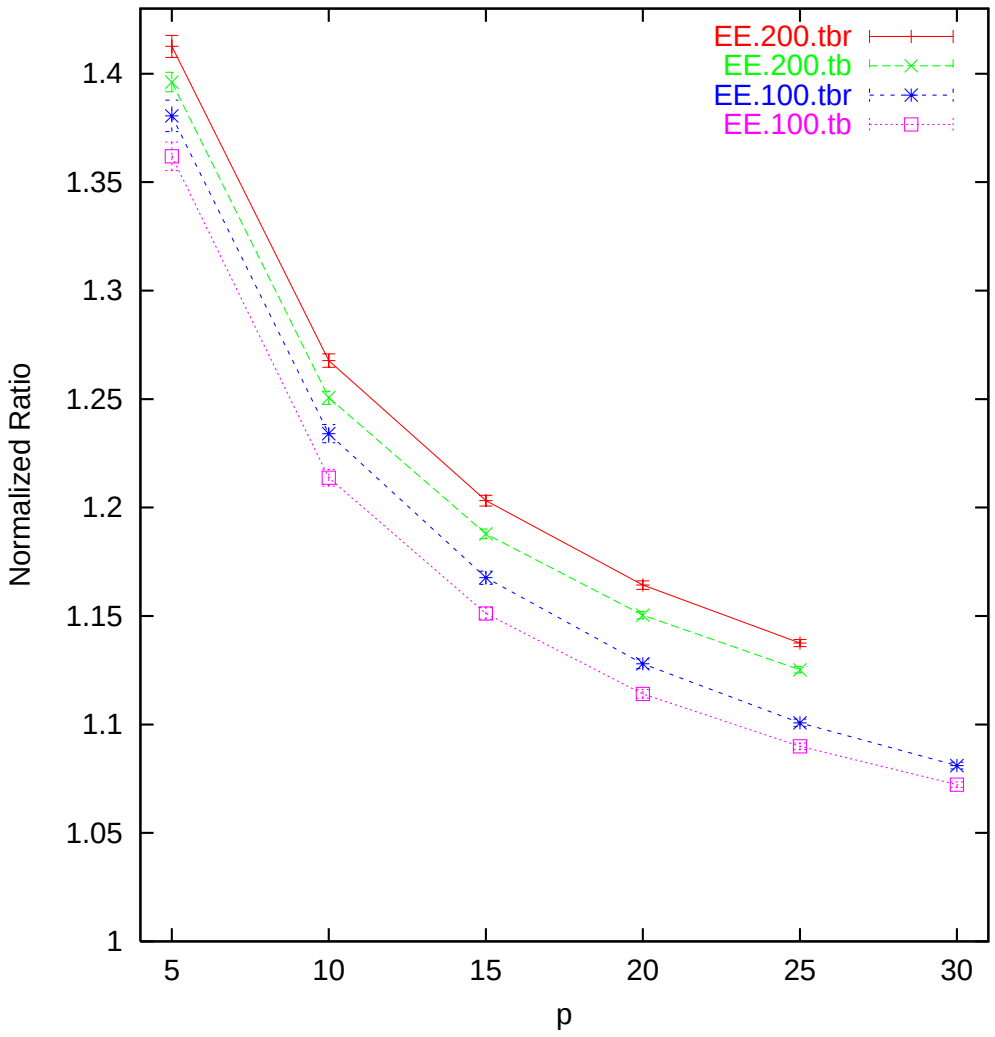


Figure 5.5: (EquallyLikely, EquallyLikely) input: The normalized ratio vs. p, for the TB and TBr heuristics, for problem instances of size $n = 100$ and $n = 200$. Each point (with 95% confidence interval) is the mean of 50 problem instances.

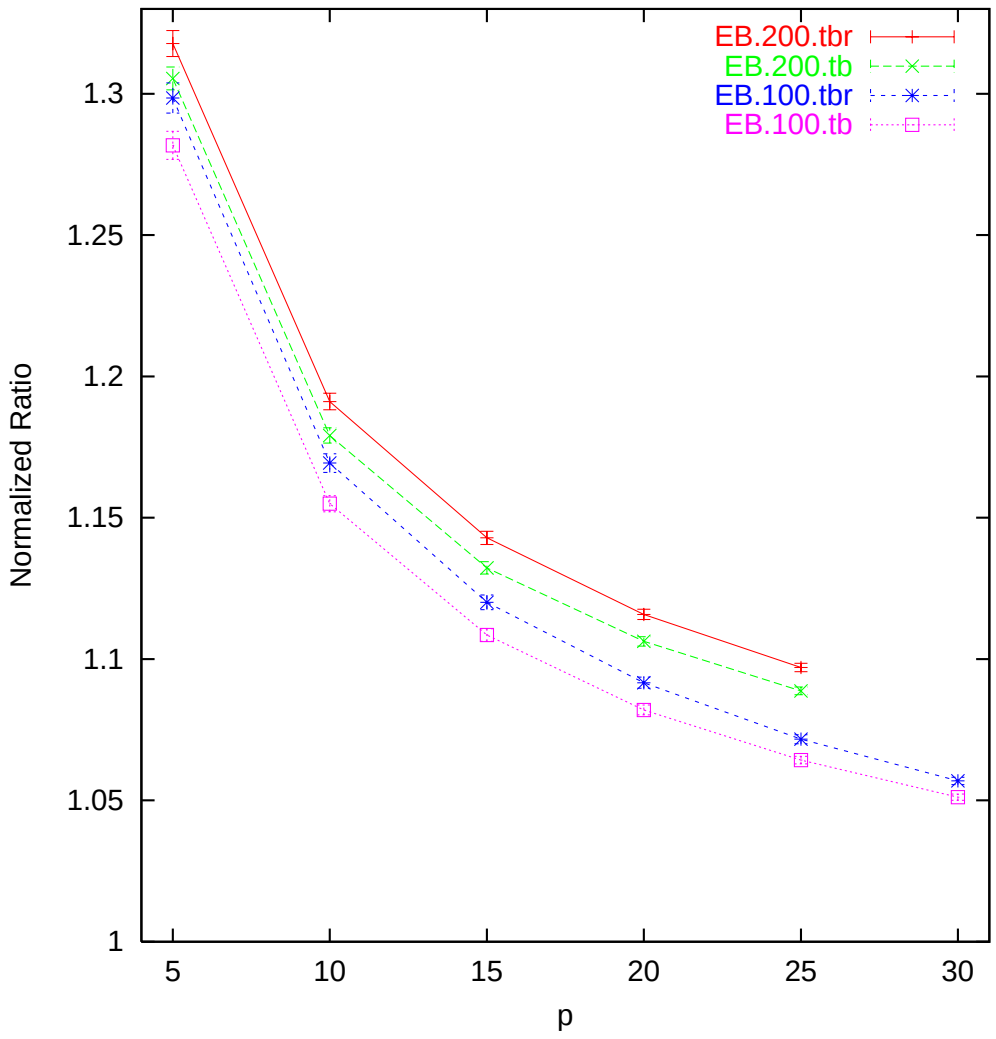


Figure 5.6: (EquallyLikely, Bimodal) input: The normalized ratio vs. p, for the TB and TBr heuristics, for problem instances of size $n = 100$ and $n = 200$. Each point (with 95% confidence interval) is the mean of 50 problem instances.

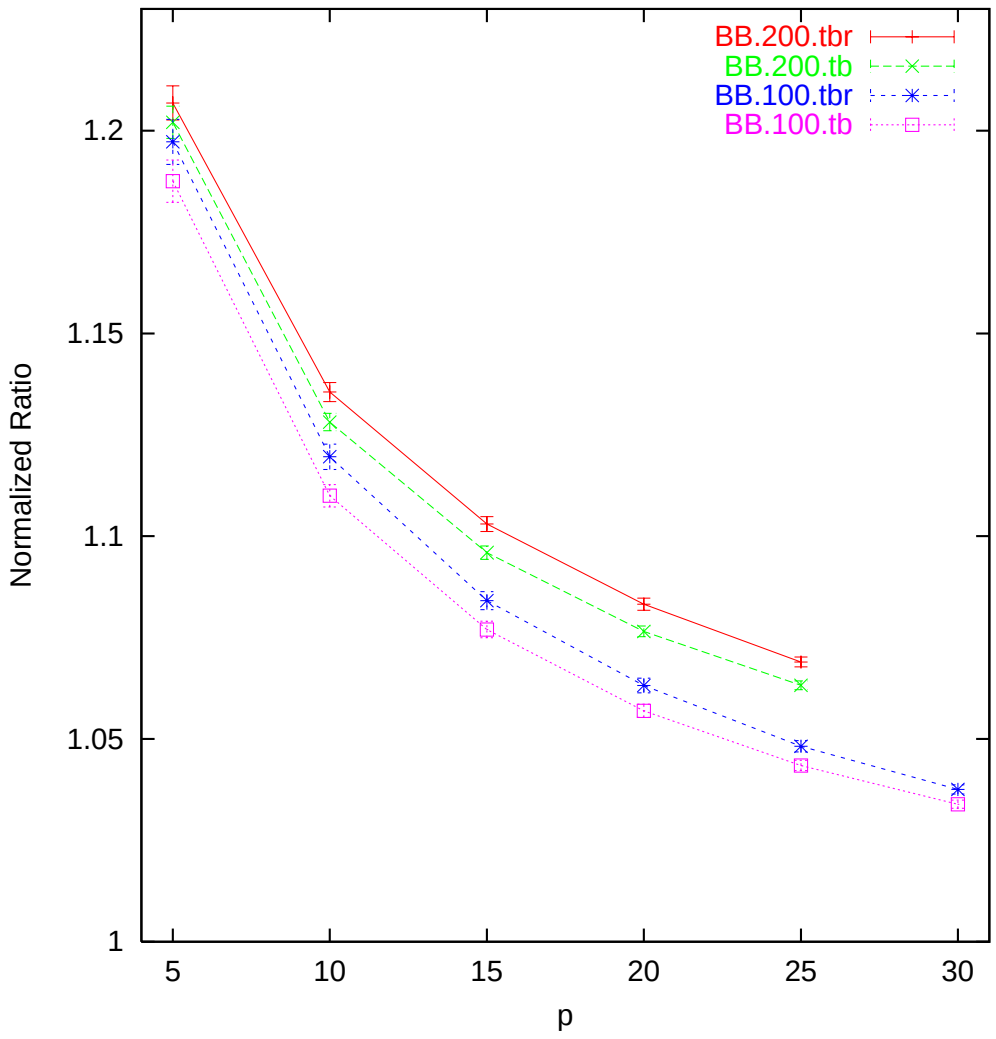


Figure 5.7: (Bimodal, Bimodal) input: The normalized ratio vs. p, for the TB and TBr heuristics, for problem instances of size $n = 100$ and $n = 200$. Each point (with 95% confidence interval) is the mean of 50 problem instances.

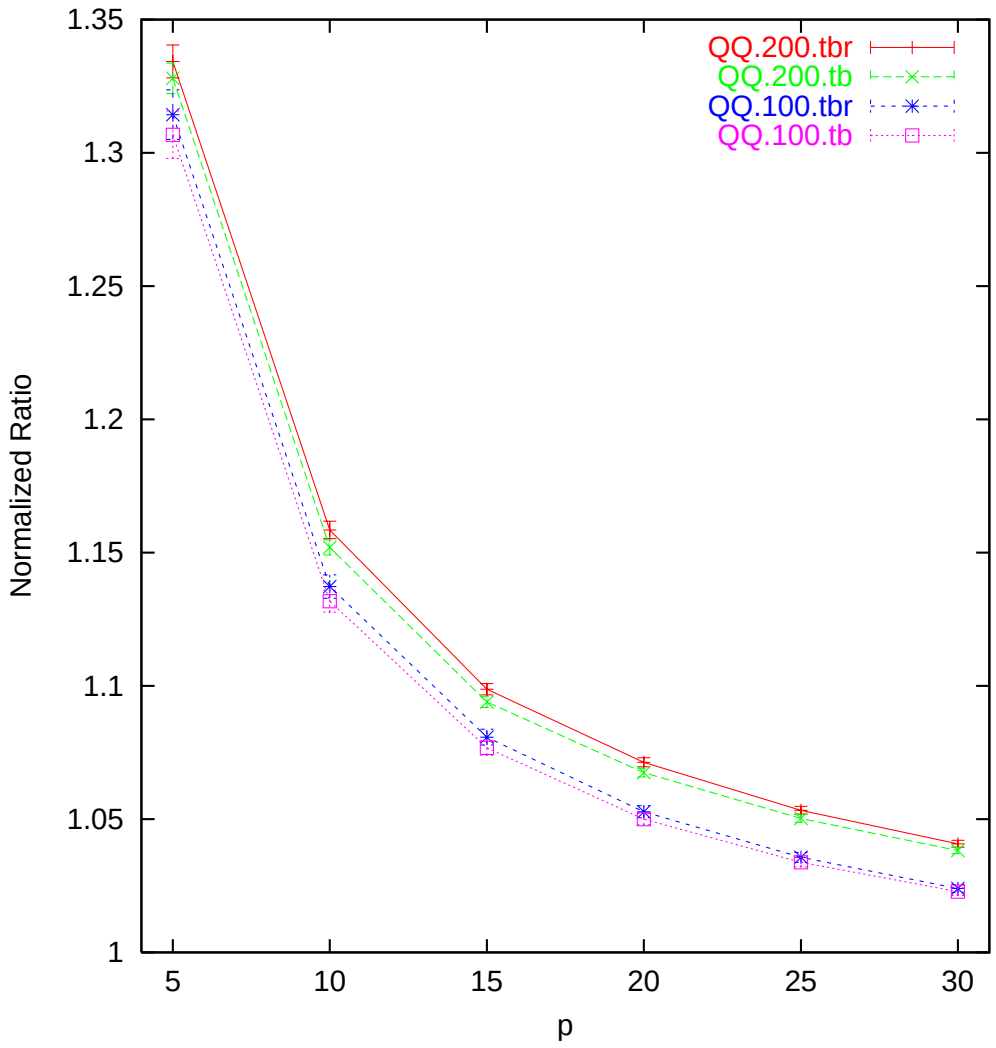


Figure 5.8: (Quadrimodal, Quadrimodal) input: The normalized ratio vs. p, for the TB and TBr heuristics, for problem instances of size $n = 100$ and $n = 200$. Each point (with 95% confidence interval) is the mean of 50 problem instances.

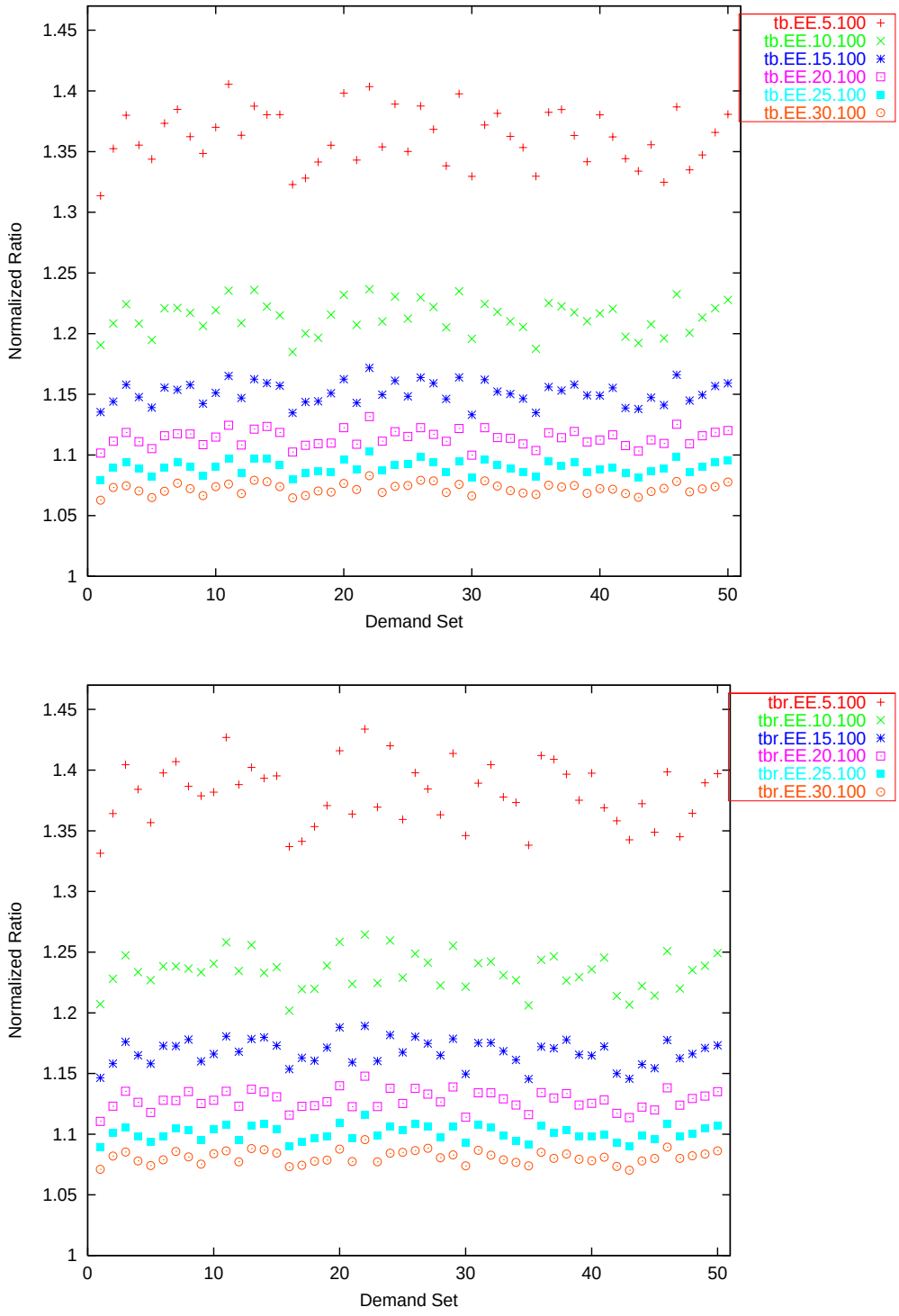


Figure 5.9: Level curves in p for the (EquallyLikely, EquallyLikely) input combination, $n = 100$. Top: TB. Bottom: TBr.

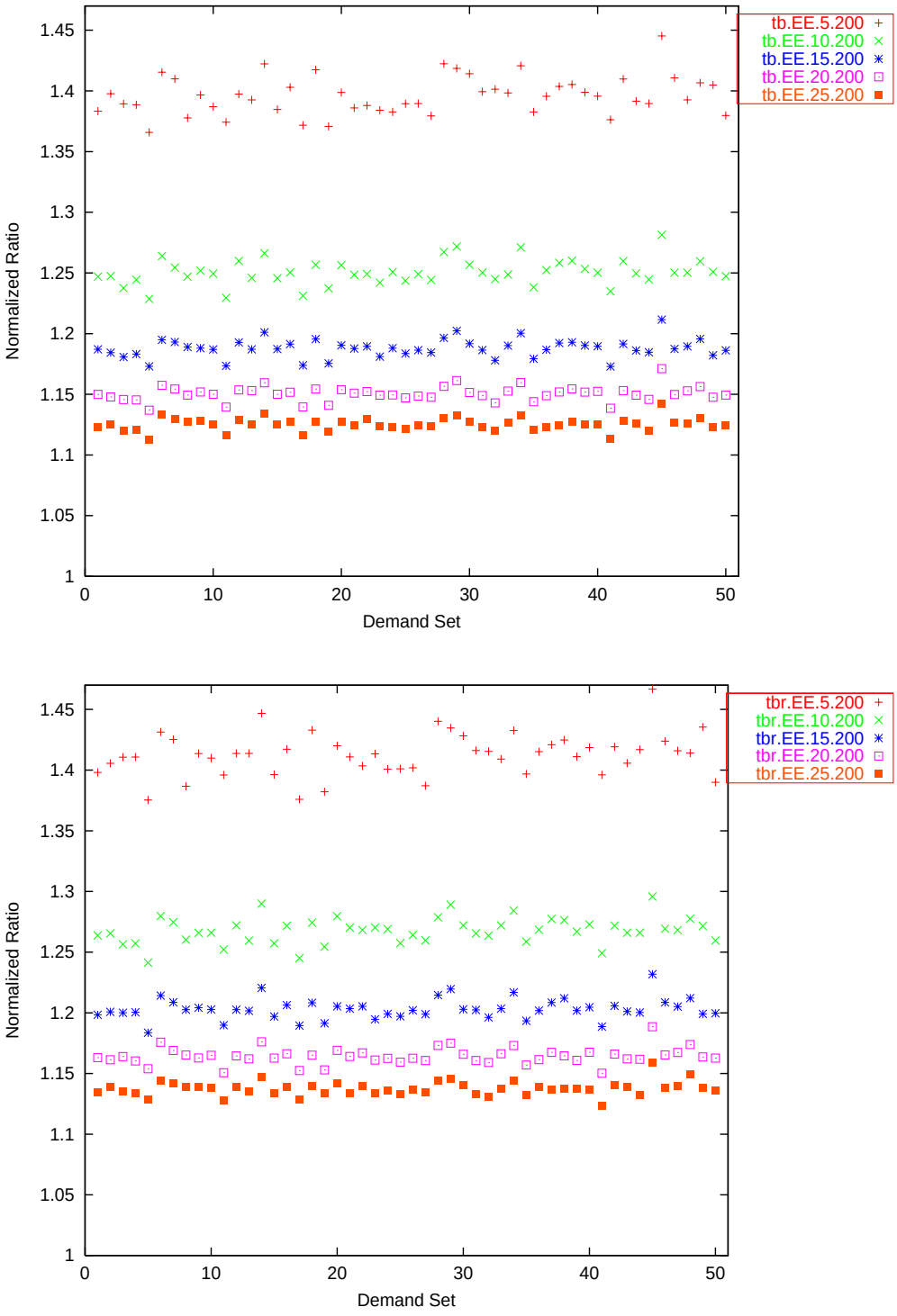


Figure 5.10: Level curves in p for the (EquallyLikely, EquallyLikely) input combination, $n = 200$. Top: TB. Bottom: TBr.

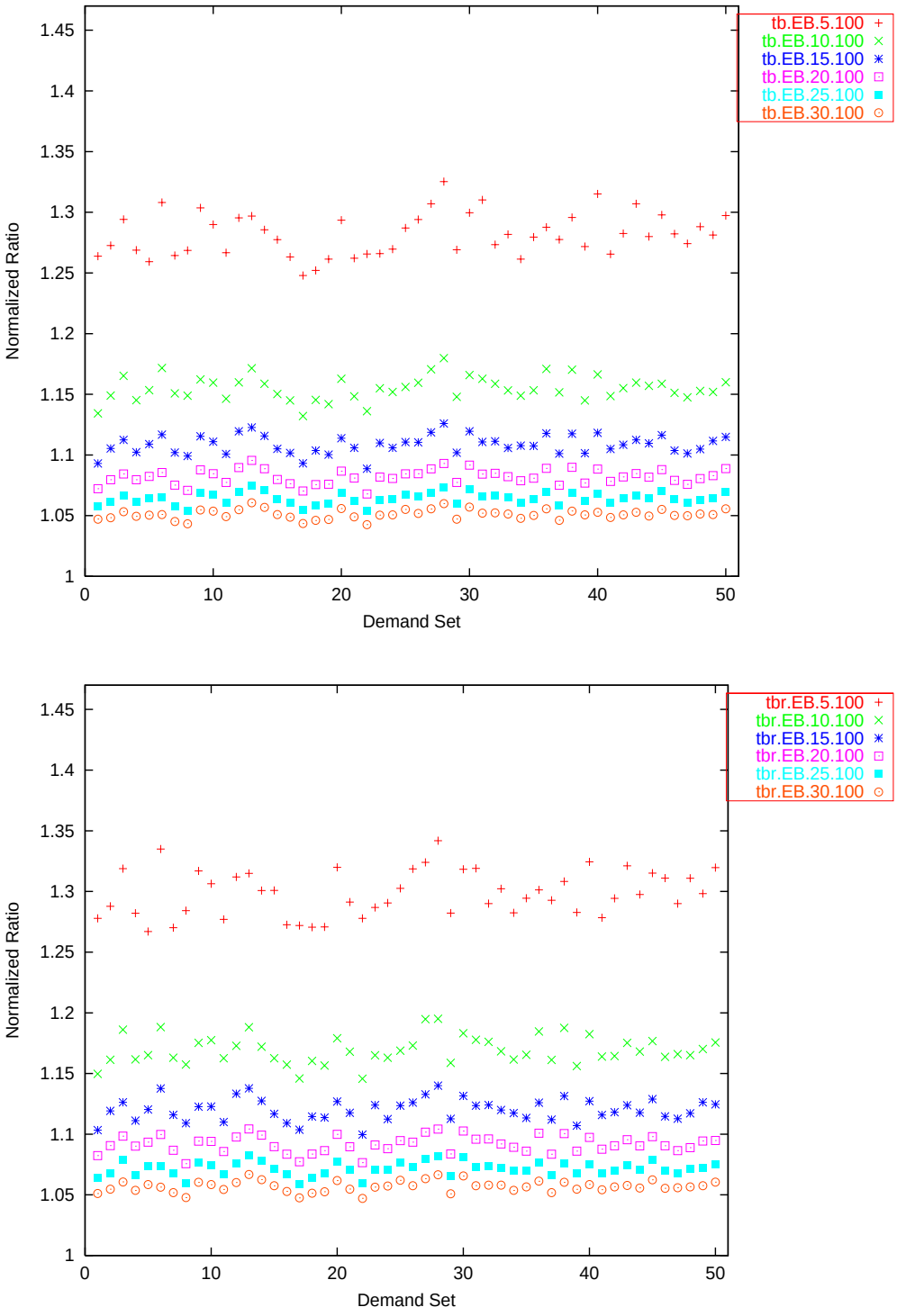


Figure 5.11: Level curves in p for the (EquallyLikely, Bimodal) input combination, $n = 100$. Top: TB. Bottom: TBr.

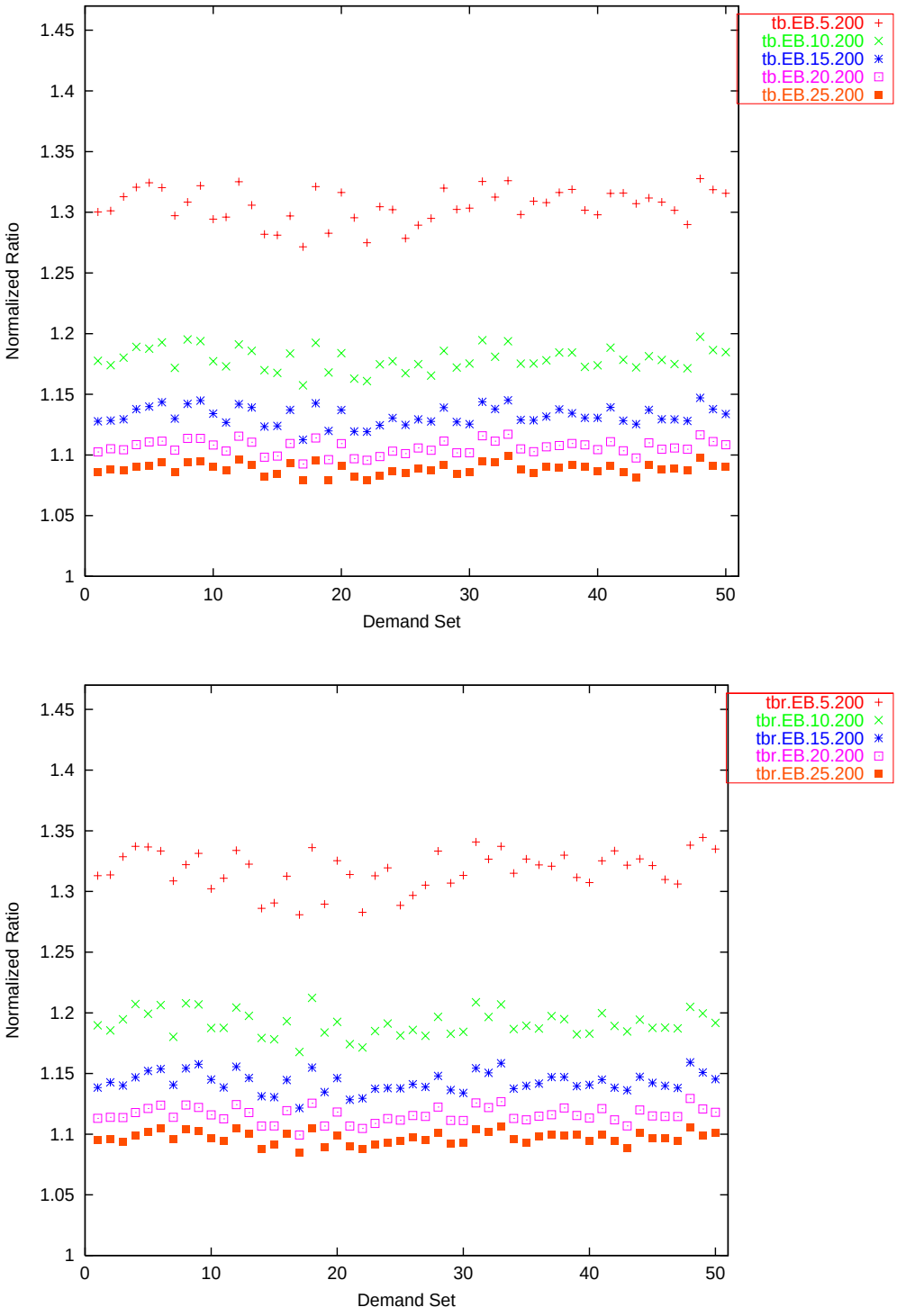


Figure 5.12: Level curves in p for the (EquallyLikely, Bimodal) input combination, $n = 200$. Top: TB. Bottom: TBr.

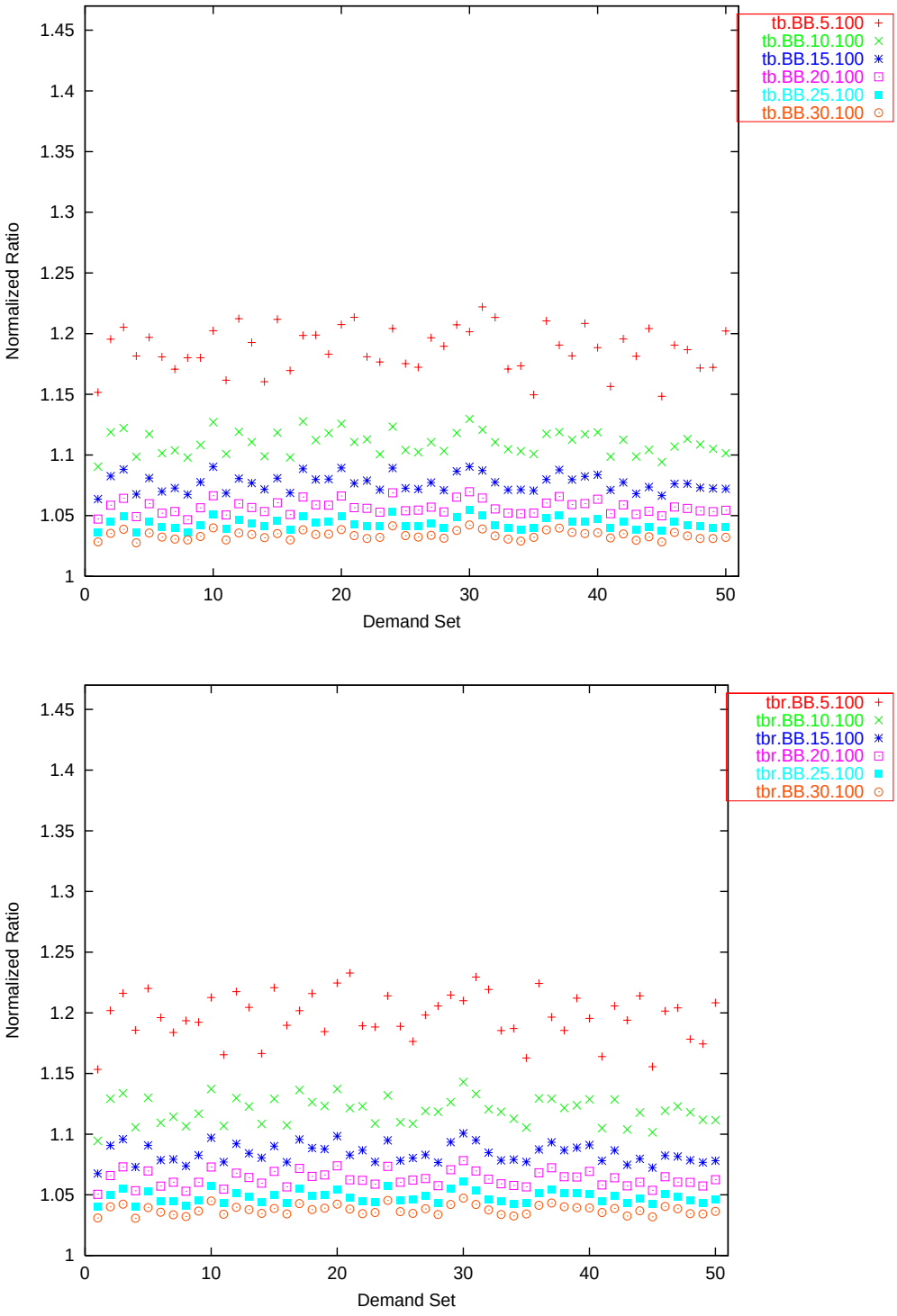


Figure 5.13: Level curves in p for the (Bimodal, Bimodal) input combination, $n = 100$. Top: TB. Bottom: TBr.

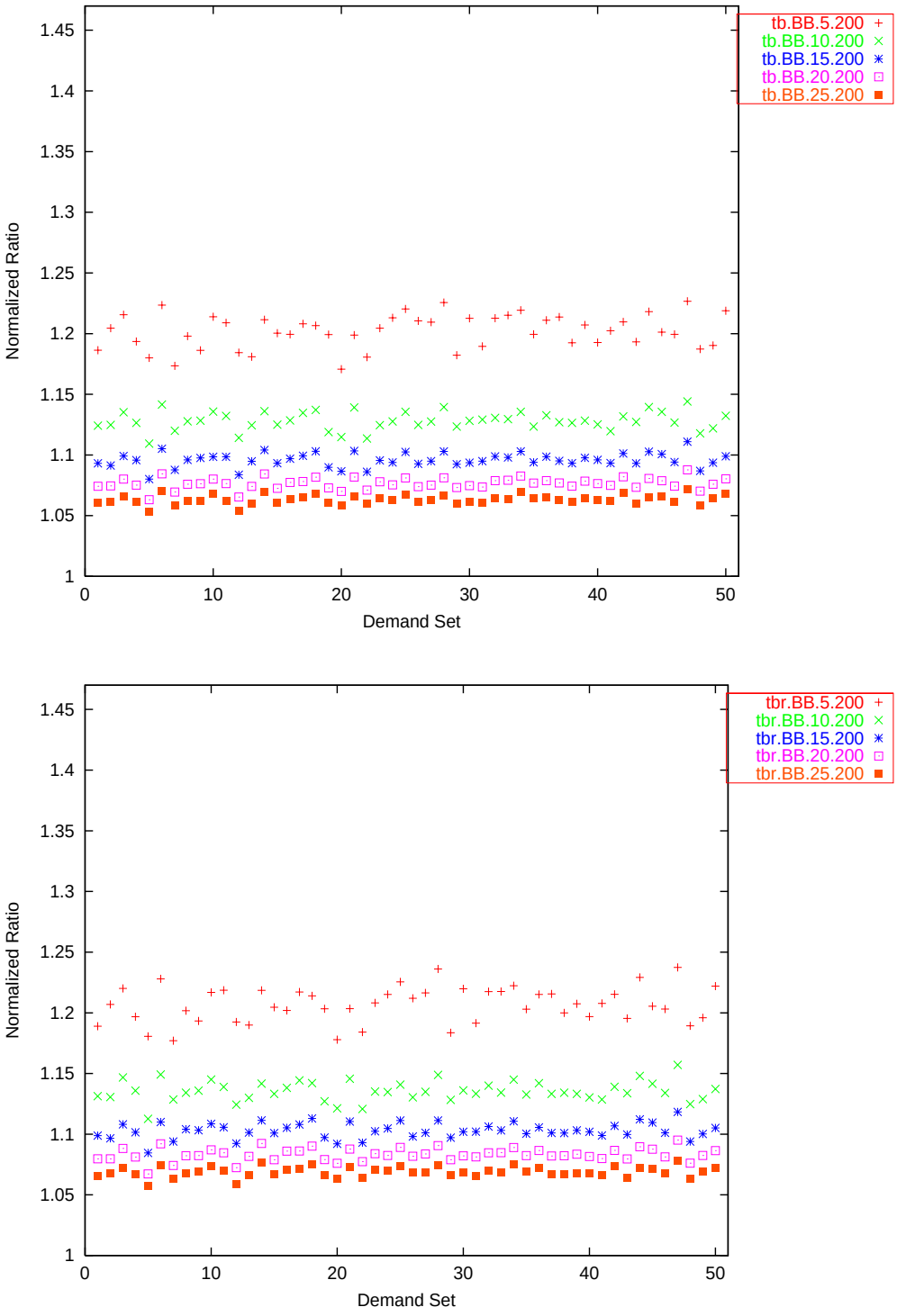


Figure 5.14: Level curves in p for the (Bimodal, Bimodal) input combination, $n = 200$. Top: TB. Bottom: TBr.

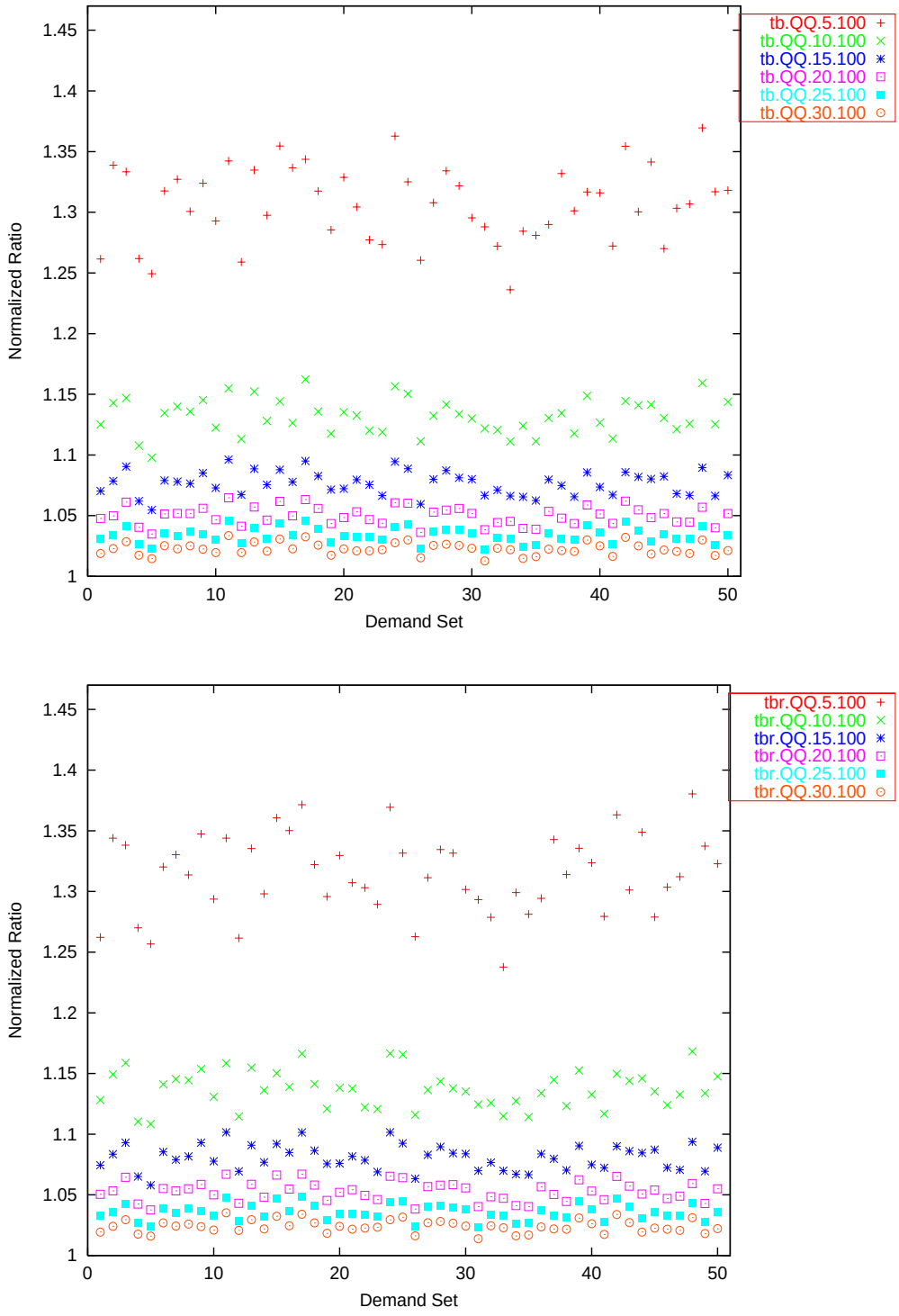


Figure 5.15: Level curves in p for the (Quadrimodal, Quadrimodal) input combination, $n = 100$. Top: TB. Bottom: TBr.

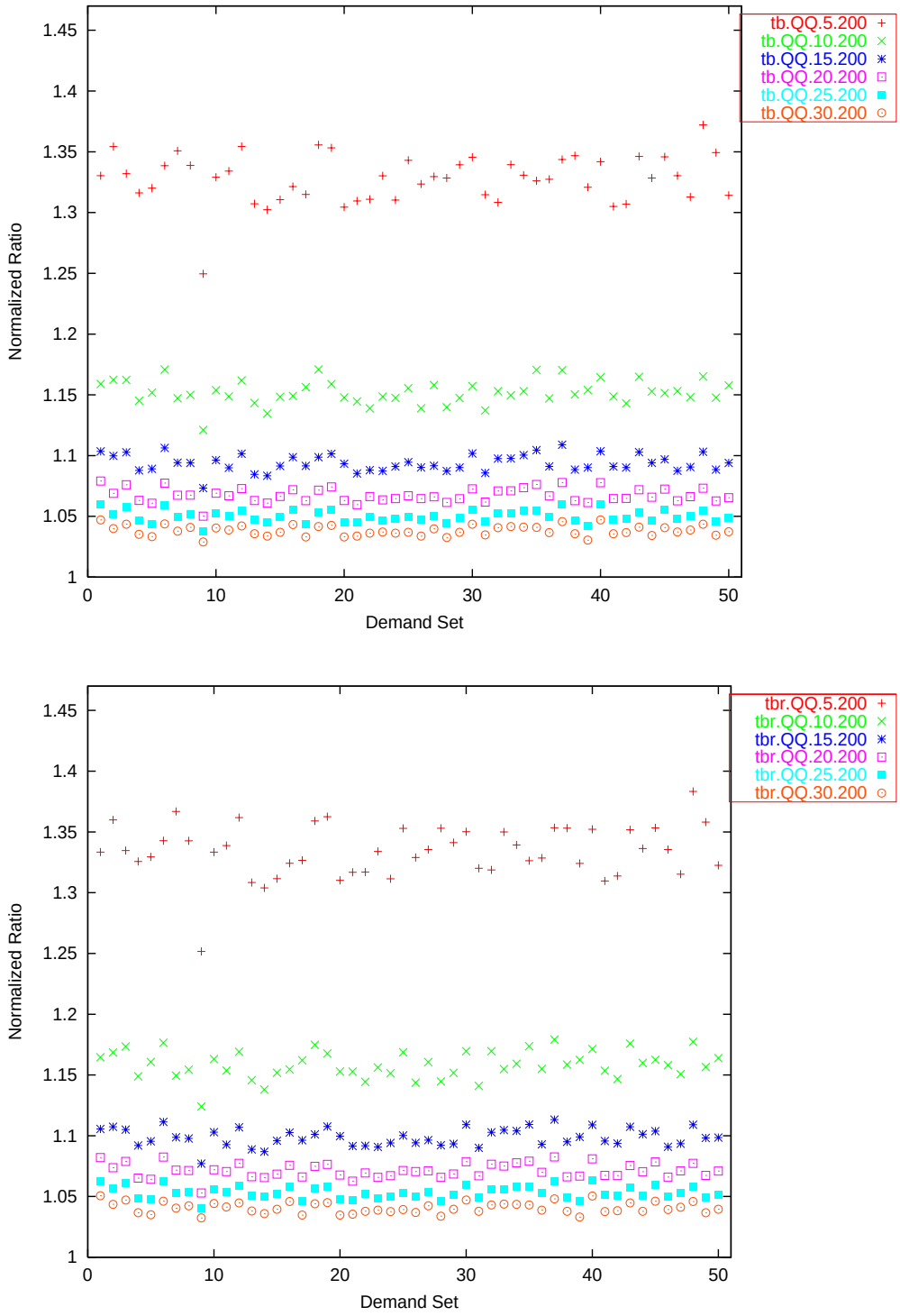


Figure 5.16: Level curves in p for the (Quadrimodal, Quadrimodal) input combination, $n = 200$. Top: TB. Bottom: TBr.

Chapter 6

Summary and future work

6.1 Summary

The traditional p-median problem asks us to find, for a given set of n demand points, the set of p supply points that minimizes the total distance of each demand point to its nearest supply point. Hassin & Tamir [13] give a $O(np)$ algorithm to solve the p-median problem on the real line, and Megiddo and Supowit [17] prove that rectilinear and Euclidean versions of the p-median problem in the plane are NP-complete.

In this thesis, we explore the p-median problem under a new distance measure, the directional rectilinear distance. On the real line, this restriction requires that the nearest supply point for a given demand point be located to the right of it, while in the plane, the nearest supply point for a given demand point must lie above and to the right of it. In general, the rectilinear c-directional, d-dimensional p-median problem forces a supply point to achieve or exceed the values of the first c coordinates of its assigned demand points.

The directional p-median problem has applications to multiprocessor

scheduling of periodic tasks as well as to traffic quantization and Quality of Service scheduling in packet-switched computer networks. For the multiprocessor application in particular, we simplify the periodic tasks scheduling algorithm from [2] by limiting it to a much smaller set of allowed service rates (supply points). While quantization has the disadvantage of requiring more resources (*e.g.*, processor time) than a continuous-rate system to accommodate a given set of customer requests, we feel that the tradeoff is more than paid for by the resulting gains in speed and simplicity.

We show that the directional p-median problem is polynomially solvable in one dimension and give two algorithms. We prove the problem NP-complete in two or more dimensions and then present an efficient heuristic to solve it. Compared to the robust Teitz & Bart heuristic for p-median, our heuristic enjoys substantial speedup while sacrificing little in terms of solution quality, making it an ideal choice for our target applications that have thousands of demand points.

6.2 Future directions

The directional Euclidean p-median problem. We expect that the NP-completeness proof in Chapter 4 is adaptable to the p-median problem under the directional Euclidean distance metric. Once the proof elements of circuits and clause configurations are specified, the remainder of the proof should follow easily.

Stochastic directional p-median in two dimensions. Analogous to problem SDPM1 in Section 3.5, we could consider the directional problem of finding the optimal set of p supply points for a given joint probability density function

describing the population of demand points (x_i, y_i).

Benefits to other scheduling algorithms. As with the PD^2 algorithm for scheduling periodic tasks on multiple identical processors, we believe that other scheduling algorithms can benefit from having quantized input. The popular packet scheduling algorithm Weighted Fair Queueing (WFQ) is a good candidate.

Further testing of TB Restricted heuristic. It would be worthwhile to evaluate the TB Restricted heuristic on DPM2 problem instances that are harder than most, for example, instances for which most of the runs of the Teitz & Bart heuristic fail to find the optimal solution. In addition, TBr could readily be extended to the 3-dimensional directional p-median problem.

Bibliography

[1] Alok Aggarwal, Baruch Schieber, and Takeshi Tokuyama. Finding a minimum weight k-link path in graphs with Monge property and applications. Discrete Computational Geometry, 12:263–280, 1994.

[2] James H. Anderson and Anand Srinivasan. A new look at Pfair priorities. Technical report, University of North Carolina, September 1999.

[3] James H. Anderson and Anand Srinivasan. Mixed Pfair/ERfair scheduling of asynchronous periodic tasks. Journal of Computer and System Sciences, 2001. Submitted.

[4] S. K. Baruah, N. K. Cohen, C. G. Plaxton, and D. A. Varvel. Proportionate progress: A notion of fairness in resource allocation. Algorithmica, 15(6):600–625, 1996.

[5] Mark Daskin. Network and Discrete Location: Models, algorithms, and applications. John Wiley and Sons, New York, 1995.

[6] Paul J. Densham and Gerard Rushton. A more efficient heuristic for solving large p-median problems. Papers in Regional Science: The Journal of the RSAI, 71(3):307–329, 1992.

[7] M. L. Dertouzos and A. K-L. Mok. Multiprocessor on-line scheduling of hard-real-time tasks. IEEE Transactions on Software Engineering, 15(12):1497–1506, Dec 1989.

[8] I. Dhillon. A new algorithm for the symmetric tridiagonal eigenvalue-eigenvector problem. PhD thesis, University of California, Berkeley, 1997.

[9] G. Di Battista, P. Eades, R. Tamassia, and I. G. Tollis. Graph Drawing: Algorithms for the Visualization of Graphs. Prentice Hall, 1999.

[10] Zvi Drezner and Horst W. Hamacher. Facility location: Applications and theory. Springer Verlag, Berlin, 2002.

[11] Zvi Drezner, Kathrin Klamroth, Anita Schoebel, and George O. Wesolowsky. The Weber problem. In Zvi Drezner and Horst W. Hamacher, editors, Facility location: Applications and theory, pages 1–36. Springer Verlag, Berlin, 2002.

[12] Michael R. Garey and David S. Johnson. Computers and Intractability: A Guide to the Theory of NP-Completeness. W.H. Freeman and Company, New York, 1979.

[13] R. Hassin and A. Tamir. Improved complexity bounds for location problems on the real line. Operations Research Letters, 10:395–402, 1991.

[14] H.W. Kuhn. On a pair of dual nonlinear programs. In J. Abadie, editor, Nonlinear programming, pages 39–54. North-Holland Publishing Company, Amsterdam, 1967.

[15] C. Lea and A. Alyatama. Bandwidth quantization and states reduction in the broadband ISDN. IEEE/ACM Transactions on Networking, 3:352–360, June 1995.

[16] David Lichtenstein. Planar formulae and their uses. SIAM Journal on Computing, 11(2):329 – 343, 1982.

[17] Nimrod Megiddo and Kenneth J. Supowit. On the complexity of some common geometric location problems. SIAM Journal on Computing, 13(1):182–196, February 1984.

[18] Bernard M. Moret. The Theory of Computation. Addison-Wesley, Reading, MA, 1998.

[19] L. Ostresh. The multifacility location problem: Applications and descent theorems. Journal of Regional Science, 17:409–419, 1977.

[20] A. Papakostas and I. G. Tollis. Algorithms for area-efficient orthogonal drawings. Computational Geometry: Theory and Applications, 9:83 – 110, 1998.

[21] C.S. ReVelle. Facility siting and integer-friendly programming. European Journal of Operational Research, 65:147, 1993.

[22] C.S. ReVelle and R.W. Swain. Central facilities location. Geographical Analysis, 2:30–42, 1970.

[23] K.E. Rosing, E.L. Hillsman, and H. Rosing-Vogelaar. A note comparing optimal and heuristic solutions to the p-median problem. Geographical Analysis, 11:86–89, 1979.

[24] K.E. Rosing and C.S. ReVelle. Heuristic concentration: two stage solution construction. European Journal of Operational Research, pages 75–86, 1997.

[25] D.A. Schilling, K.E. Rosing, and C.S. ReVelle. Network distance characteristics that affect computational effort in p-median location problems. European Journal of Operational Research, 127:525–536, 2000.

[26] M.B. Teitz and P. Bart. Heuristic methods for estimating the generalized vertex median of a weighted graph. Operations Research, 16:955–961, 1968.

[27] Alfred Weber. Ueber den Standort der Industrien, 1909, translated as Theory of the Location of Industries. University of Chicago Press, Chicago, 1929.

[28] Mark Ziegelmann. Constrained Shortest Paths and Related Problems. PhD thesis, Universitaet des Saarlandes, 2001.

Appendix A

Algorithm Quantize for DPM1

```
{* fill in Diff table * }
for i = 1 to n do
  for j = i to n do
    Diff[i][j] = x[j] − x[i] ;
  end for
end for
{* fill in Cumul table * }
for i = 1 to n do
  for j = i to 1 do
    if j == i then
      Cumul[i][j] = 0
    else
      Cumul[i][j] = Diff[j][i] + Cumul[i][j + 1]
    end if
  end for
end for
{* build Opt table *}
for i = 1 to n do
  for j = 1 to i do
    if i == j then
      Opt[i][j]= 0
      Prev[i][j] = i − 1
    else if j == 1 then
      Opt[i][j] = Cumul[i][1]
```

```
      else
        Opt[i][j] = min_{j-1≤s≤i-1}{Opt[s][j − 1] + Cumul[i][s + 1] }
        Prev[i][j] = s_min
      end if
    end for
end for
{* construct the z array from Prev *}
z[p] = x[n]
index[p] = n ;
for i = p − 1 to 1 do
    index[i] = Prev[index[i + 1]][i + 1]
    z[i] = x[index[i]]
end for
```

Appendix B

Algorithm Quantize-Continuous for SDPM1

```
{* fill in Sum table (analogous to Diff) * }
for i = 1 to n do
  for j = i to n do
    if j == i then
      Sum[i][j] = m[i]
    else
      Sum[i][j] = Sum[i][j − 1] + m[j]
    end if
  end for
end for
{* fill in Prod table (analogous to Cumul) * }
for i = 1 to n do
  for j = 1 to i do
    Prod[i][j] = Sum[j][i] × e[i]
  end for
end for
{* build AQL table (analogous to Opt) *}
for i = 1 to n do
  for j = 1 to i do
    if i == j then
      AQL[i][j] = AQL[i − 1][j − 1] + Prod[i][j]
```

```
      Prev[i][j] = i − 1
    else if j == 1 then
      AQL[i][j] = Prod[i][1]
    else
      AQL[i][j] = min_{j−1≤s≤i−1}{AQL[s][j − 1] + Prod[i][s + 1] }
      Prev[i][j] = s_min
    end if
  end for
end for
{* construct the z array from Prev *}
z[p] = e[n]
index[p] = n ;
for i = p − 1 to 1 do
  index[i] = Prev[index[i + 1]][i + 1]
  z[i] = e[index[i]]
end for
```

Appendix C

Algorithm Q-PD^2 for scheduling a quantized task set

```
{* Pre-Processing phase * }
Q = BuildQueues(x)
t = 0
{* Start of Scheduling phase * }
while true do
  repeat
    T = ExtractMin(Q)
    Schedule task T in slot t
    t_next = the earliest future time at which T will be eligible again
    T.nextEligible = t_next ;
    T.priority = Determine T's priority at time t_next ;
    Enqueue(Q,T)
  until m tasks have been scheduled in slot t
  t = t + 1
end while
```