Noah's Bagels - Some Combinatorial Aspects

Hadas Shachnai and Tami Tamir*

Department of Computer Science,
The Technion, Haifa 32000, Israel.

Abstract

We discuss some of the problems Noah had to face when he established his famous bagel business; Noah had to decide how many bagels of each flavor should be produced in each of his bakeries, subject to space, production and flavor popularities constraints. Noah's goal was to find solutions that maximize his revenue (the Revenue problem) while guaranteeing satisfaction to his customers (the Balancing problem). Noah showed that both problems are NP-Hard. He then looked at some special cases, where each of these problems can be solved optimally in polynomial time. Finally, he found a nearly optimal solution for the Revenue problem that paved his road to success.

*contact e-mail: tami@cs.technion.ac.il

1 Introduction

This is the amazing story of Noah's Bagels: we discuss some of the problems Noah had to face when he established his bagel business. Naturally, Noah wished to maximize his revenue, while guaranteeing satisfaction to his customers. Therefore he needed to solve some *balancing problems*, that arose due to space and production constraints in each of his bakeries. Later, Noah realized that, in fact, he was dealing with an interesting version of the classic *bin packing* problem. Moreover, the solutions he found for his baking problems can also be applied to some resource allocation problems arising in multimedia systems. We elaborate on that below.

When Noah started his famous bagel business, he first sold only 'plain' bagels, made of the dough he developed using his secret formula. After several successful weeks, the customers were looking for innovations, and he decided to extend the variety of his products and to produce bagels of different flavors (such as onion, sesame, blueberry, raisin and many more). All bagels had the same size and were made from the same dough. After a few more successful weeks, Noah couldn't stand the demand and he decided to purchase and operate additional bakeries.

In his new bakeries Noah planned to produce a large variety of bagels. Each bakery can produce daily a limited number of bagels, depending on the size of the oven and the amount of secret dough that can be produced in it. In addition, being very neat and fastidious, Noah insisted on storing the ingredients for each flavor in a separate cabinet, so due to space limitation, each bakery had also a limited number of different flavors that could be baked in it.

The fresh bagels were delivered every day from the bakeries to a new fancy store, which was, by that time, very popular and famous [1].

Few weeks passed and the customers began to complain: "You always run out of sesame bagels" said one. "Whenever I get to the store, I can only find huge baskets of blueberry bagels, and I have to wait for my favorite onion bagels" said another. Noah was very embarrassed. He realized that some flavors are more popular than the others, and since he was a former mathematician he decided to examine the problem more carefully.

Noah knew that the amount of bagels baked of each flavor must be determined by its popularity. After collecting some statistics, Noah found out the popularity of each of the flavors. Next, he had to decide how many bagels of each flavor should be baked in each bakery, in order to reduce the number of unsatisfied customers. "If the customers have their favorite bagel available and fresh, I will be able to sell all the bagels produced in my bakeries", he thought.

We now turn to a more formal description of Noah's problem. Denote by $\{b_1, b_2, \ldots, b_N\}$ the N bakeries operated by Noah, and by $\{f_1, f_2, \ldots, f_F\}$ the F flavors. Let Q_j denote the maximal number of bagels that can be produced daily in b_j, and let C_j denote the number of cabinets in b_j. Clearly, the maximal number of bagels that can be produced daily is $Q = Q_1 + \ldots + Q_N$.

The popularity of the flavor f_i is denoted by $p_i \in [0, 1]$, such that $\sum_{i=1}^{F} p_i = 1$. For each bakery b_j and flavor f_i, Noah had to decide how many bagels of flavor f_i will be produced in b_j. Formally, Noah was looking for two $F \times N$ matrices: the *cabinet matrix*, A, $A_{i,j} = 1$ iff the ingredients for f_i are stored in b_j, and the *bagel allocation matrix*, B, where $B_{i,j}$ denotes how many bagels of flavor f_i are baked in b_j. We denote by B_i the total number of bagels produced of flavor f_i, namely, $B_i = B_{i,1} + \ldots + B_{i,N}$.

The matrices B and A must satisfy the following conditions:

1. $A_{i,j} = 0 \Rightarrow B_{i,j} = 0$.

2. $\forall 1 \leq j \leq N$, $\sum_i B_{i,j} \leq Q_j$.

3. $\forall 1 \leq j \leq N$, $\sum_i A_{i,j} \leq C_j$.

Noah's first optimization problem was the *Revenue* problem:

> Given N, F, $p_1, \ldots, p_F$, $Q_1, \ldots, Q_N$, and $C_1, \ldots, C_N$, find assignment matrices A, B, such that $\sum_{i=1}^{F} Min(p_i \cdot Q, B_i)$ is maximized.

Noah's second optimization problem was the *Balancing* problem:

> Given N, F, $p_1, \ldots, p_F$, $Q_1, \ldots, Q_N$, and $C_1, \ldots, C_N$, find assignment matrices A, B, such that $\forall 1 \leq i \leq F$, $B_i \geq (1-\varepsilon)p_i \cdot Q$, and ε is minimized.

Noah was looking for *a perfect assignment.* Such assignment achieves both the maximal possible revenue, Q, and is optimally balanced, that is $\varepsilon = 0$.

Before he started to look for solutions to his assignment problems, Noah decided to examine some related work.

1.1 On Bagels, Bin-Packing and Video Programs

Noah noticed that he was actually dealing with a special version of the well known bin-packing problem. It turned out that his problem is also relevant to resource allocation in Multimedia-On-Demand (MOD) computer systems:

A MOD system consists of a central processor and a collection of N shared disks. MOD systems should be able to "play" multiple streams of different video programs simultaneously, based on customer demand. There are F different video program files $\{f_1, f_2, \ldots, f_F\}$; the popularity of each program is known and given by $p_i \in [0,1]$ such that $\sum_{i=1}^{F} p_i = 1$; disk j has *broadcast capacity* Q_j, which specifies the maximum number of concurrent video streams that can be transmitted by disk j, and *storage capacity* C_j, which specifies the number of different video program files that can be stored on disk j.

As in the bagel production problem, the goal is to assign copies of the programs on the disks in a way that reflects their popularities and maximizes the total number of possible broadcasts at any given time.

1.2 Previous Work

Noah considered first related work on multimedia systems. He noticed that earlier studies (e.g. [7]) used a different measure for balancing, which considers only the *number of cabinets* allocated to each flavor and ignores the number of bagels produced of this flavor. Formally, the goal in these works is that $\sum_j A_{i,j}$ will reflect the popularity of the flavor f_i.

Noah observed that this criterion can yield poor results. Popular flavors may not be produced in their desired amount if they are baked in many *small* bakeries. "My real goal", he thought, "is that $\sum_j B_{i,j}$, the *number of produced bagels*, will reflect the flavor popularity".

The assignment problem reminded Noah of the classic *Bin-Packing* problem [3]: each bakery is a bin and the purpose is to 'pack' the flavors into the bins. However, in the original bin-packing problem all the bins are identical and there is no limit on the number of elements that can be stored in a bin. Noah found some related work that deals with variants of bin-packing, like [6], that considers a model in which no bin can contain more than n elements and [2], where bins of different sizes are allowed. "My problem is yet another version of Bin-Packing", Noah thought. "In this new version (i) there are bounds on the bin sizes, and on the number of elements that can be stored in each bin, (ii) the bins are not identical, and (iii) elements can be split among several bins (e.g. I can bake onion bagels in more than one bakery). In the classic Bin-Packing problem each element must go to exactly one bin".

Another related work [4] deals with the *multiprocessor scheduling* problem, where m jobs have to be assigned to n processors, so as to minimize the overall completion time of the schedule. The paper presents a 4/3-approximation algorithm. As in the bin-packing case, this algorithm could not be adopted by Noah.

1.3 Noah's Results

The assignment problems bothered Noah for a while; he decided to quit his bagel business and to dedicate his life to the study of these problems. We are very lucky to publish his adventures and research results in this paper. We now summarize Noah's results as given in the next sections:

- The problem of finding the optimal assignment is NP-Hard. This is true for both the balancing and the revenue problems.
- For some instances a perfect assignment always exists and can be found in polynomial time. Three simple conditions for the existence of a perfect assignment are given. For each condition, Noah showed how to find the optimal solution when this condition is satisfied.
- When the conditions are not met, Noah showed how to approximate a perfect assignment. The approximation ratio depends on the uniformity of the bakeries. Formally, given r and $\alpha \geq 1$ such that $\forall j,\ r \leq \frac{Q_j}{C_j} \leq \alpha \cdot r$, Noah found an algorithm that achieves $\varepsilon = \frac{\alpha - 1}{\alpha}$ for the balancing problem and revenue $\frac{1}{\alpha}Q$. If cabinets are allowed to be moved from one bakery to another, Noah showed that the above α approaches 1.
- Noah showed that by adding *a single* cabinet (i.e. one storage unit) to each bakery, the Revenue problem can be solved optimally in polynomial time.

The hardness results are given in Section 2.4. Section 3 considers the cases where perfect assignment exists, can be found, and approximated (Section 3.3.1). The nearly optimal solution for the Revenue problem is given in Section 4. In the Appendix we describe the application of the Revenue and the Balancing problems to MOD systems. Due to space limitations some of the proofs are omitted. These proofs will be given in the full version of the paper.

2 Preliminaries

Noah's main concern was to have the amount of bagels produced in each flavor proportional to its popularity. By definition, $B_i = \sum_{j=1}^{N} B_{i,j}$ is an integer, while the desired number of bagels of f_i is $p_i \cdot Q$, which is not necessarily an integer. Noah was not interested in baking bagels with mixed flavors, so he rounded the numbers $p_i \cdot Q$ according to some apportionment algorithm he once read about [5]. He denoted the rounded numbers by W_i, $1 \leq i \leq F$, and it holds that $\sum_i W_i = Q$.

2.1 Balanced Assignment

Noah's first measure for the quality of an assignment was according to the balance criterion:

Definition 2.1 *A static assignment is* c-balanced *if for every flavor* f_i, $B_i \geq c \cdot W_i$.

An optimal assignment is 1-balanced. In a 1-balanced assignment, exactly W_i bagels are baked of each flavor f_i. Note that for any assignment $c \in [0, 1]$. Noah was fascinated: he was sure that it will be very easy for him to come up with a 1-balanced assignment. He could see in his mind the happy customers coming to the store and taking their favorite bagels without waiting. Suddenly, he froze and some important questions crossed his mind: Does a 1-balanced assignment exist for any instance of the problem? Can I find it efficiently? If a 1-balanced assignment does not exist, can I find or approximate an optimal assignment efficiently?

2.2 Revenue of an Assignment

Noah thought to himself: "Even if I achieve the best possible balanced assignment, it does not necessarily maximize my revenue. I prefer, for example, to discriminate the blueberry lovers that are few, and to bake more sesame bagels; these are much more popular, and in the balanced assignment they are not produced in their desired quantity". Noah was right again, the overall potential of the system to satisfy customer requests is not necessarily achieved by a balanced assignment. Achieving the maximal *revenue* is a different goal, and sometimes it is even more desirable than achieving a balanced assignment.

Definition 2.2 *The* revenue of an assignment *is* $R = \sum_{i=1}^{F} Min(W_i, B_i)$.

Noah decided to consider the minimum of W_i and B_i because he gained nothing from baking more than W_i units of f_i. Noah bagels are known for their freshness, and no leftovers are used; if too many bagels of a particular flavor are baked, it's a waste of dough and ingredients. In calculating the revenue, we are interested only in the total number of bagels that may be required and provided. Similarly, when balancing is the goal, we gain nothing if for some flavor f_i, $B_i > W_i$.

The questions dealing with the existence of balanced assignment arise again, this time about the revenue: Can the maximal possible revenue be found? approximated?

2.3 Combining Revenue and Balancing

Noah decided to explore the relation between the revenue and the balancing problem. He noticed that a 1-balanced assignment has also the maximal possible revenue, which is $Q = \sum_i W_i$. "So that's my goal", he thought,"I have to find a *Perfect Assignment*!".

After a while Noah realized that for some instances a perfect assignment does not necessarily exist. Consider a simple instance of two bakeries, each with $C_j = 1$ and $Q_j = 10$, and two flavors with $p_1 = \frac{3}{4}$ and $p_2 = \frac{1}{4}$. A perfect assignment should enable the production of 15 bagels of the popular flavor and 5 bagels of the second. It can be seen that there is no legal assignment that satisfies these requests.

Noah understood that he should seek for the best possible assignment. However, the two goals may conflict. Consider an instance with two bakeries: $C_1 = 2$, $Q_1 = 20$; $C_2 = 1$, $Q_2 = 10$; and three flavors: sesame, onion and blueberry, with $W_1 = 14$, $W_2 = 14$, $W_3 = 2$. The assignment that yields the maximal revenue is presented in Figure 1(a). The revenue is 28. This assignment, however, is 0-balanced - the blueberry lovers do not get any portion of their requirement. Figure 1(b) presents the best possible balanced assignment in this instance. It is a $\frac{10}{14}$-balanced assignment, and its revenue is 26. Generally, any c-balanced assignment achieves revenue at least $c \cdot Q$.

2.4 Hardness of the Perfect Assignment Problem

The next question Noah considered is whether he can detect efficiently if a perfect assignment exists.

Theorem 2.1 *Given an instance of the assignment problem, it is NP-Hard to determine if a perfect assignment exists for this instance.*

Proof: We show a reduction from the *partition problem* which is known to be NP-Hard [3]. The partition problem consists of a finite set A, and size $s(a)$ for each $a \in A$. The problem is to determine if there exists a subset A' of A such that $\sum_{i \in A'} s(i) = \sum_{i \in A \setminus A'} s(i)$.

Given an instance for partition, consider the assignment problem consisting of two flavors, with equal popularities, $p_1 = p_2 = \frac{1}{2}$, and $|A|$ bakeries with $C_j = 1$ and $Q_j = s(a_j)$. For this problem, every perfect assignment induces a desired partition and vice-versa. ∎

Since any perfect assignment is both 1-balanced, and achieves revenue Q, we conclude that each of the problems is NP-Hard.

3 Finding a Perfect Assignment

3.1 A Simple Condition for the Existence of a Perfect Assignment

Theorem 3.1 *If for all the bakeries $Q_j \leq C_j$, then a perfect assignment exists, and can be found in polynomial time.*

The proof is given in the Appendix.

"Well, It's simpler than what I imagined", thought Noah, "I can return to my business. I will add many cabinets to each of my bakeries, and then I can achieve a perfect assignment". Noah called Jeppeto, his carpenter, who also designed the bakeries, and asked him to build and add the required cabinets. "I can't add so many cabinets", answered Jeppeto, "The bakeries are too small, the workers already complain that it is too crowded and dense. Also, the wood is very expensive these days, and you can't afford it. I may be able to add *one additional cabinet* to each bakery if you like, but not more than that". "Oh, you don't understand, that won't help at all, I need many cabinets", cried Noah, and the conversation ended.

Jeppeto, who - like Noah - was a former mathematician, became curious. He knew what bothered his boss. "I understand that adding a single cabinet to each bakery is not enough", he thought, "yet I have a strange feeling that it may be very significant. Anyway, I think I'll return now to my wooden puppet, this artificial intelligent project looks promising..".

3.2 A Weaker Condition

"This Jeppeto is lazy", said Noah to himself, "I must look for a weaker condition for the existence of a perfect assignment. It seems that I should consider also the *popularities* of the different flavors when I define that condition". Few more days passed before Noah came up with the new condition:

Theorem 3.2 *Let $0 < \varepsilon \leq 1$ be the maximal number such that for all the flavors $p_i \geq \frac{\varepsilon}{F}$. If for all the bakeries $C_j \geq \frac{Q_j \cdot F}{\varepsilon \cdot Q} + 1$ then there exists a perfect assignment that uses at most $F + N - 1$ cabinets.*

Proof: Consider the simple greedy algorithm that 'allocates' bagels to flavor after flavor, filling the bakeries one after the other. In other words, each bakery, b_j, allocates bagels until it's baking capacity, Q_j, is reached, then we continue the allocation from b_{j+1}, and so on. It can be seen that this algorithm terminates with a perfect assignment that uses at most $F + N - 1$ cabinets. ∎

Noah was not satisfied with his new findings, his bakeries did not fulfill even the weaker condition. He was about to give up when the phone rang..

3.3 Uniform Capacity Ratio

"Hello Noah, this is Bigbird from Sesame street", said the voice, "my friends and I here at Sesame street have heard about your assignment problem. We are really worried about the Sesame lovers, that may be discriminated, so we decided to help you". "Bigbird, I'm so happy you called", said Noah surprisingly. "I will appreciate any help, but you know, this is science..., I hope I can trust you. If you do help me I'm ready to give Cooky-monster a huge basket of sweet bagels!". "Our plan is very simple", said Bigbird, "You only have to move cabinets from some bakeries to others, this will guarantee that you will be able to find a perfect assignment".

Noah remembered what Jeppeto told him about the price of adding *new* cabinets; "moving is much cheaper than assembling new cabinets", he estimated, and asked Bigbird to explain her plan in detail. Bigbird continued: "We have a simple condition for the existence of a perfect assignment. All you need is that your bakeries will have *uniform capacity ratio.* This means that there exists a constant r such that for every $1 \leq j \leq N$, $\frac{Q_j}{C_j} = r$. There are no assumptions on the distribution of the popularities as you used in your last result. Another condition, which is likely to be met is that $\sum_{j=1}^{N} C_j \geq F + N - 1$."

Bigbird carried on: "If you are interested, I can e-mail to you the formal proof right away. Erni and Bert have just finished to go over the final version". "Please do so", said Noah impatiently, while turning on his laptop. Few seconds passed and the following note arrived:

Theorem 3.3 (Bigbird) *If the capacity ratio is uniform and $\sum_{j=1}^{N} C_j \geq F + N - 1$ then a perfect assignment exists and can be found in polynomial time.*

Proof: We present a polynomial time algorithm that terminates with a perfect assignment. The algorithm, denoted by $\mathcal{A}_u$, proceeds bakery after bakery, for each bakery b_j, it decides on production of exactly Q_j bagels of at most C_j different flavors. We assume that the flavor requests are given in a non-decreasing order of their popularities, i.e., $W_1 \leq W_2 \leq, \ldots, \leq W_F$.

We keep the remaining requests in a sorted list, denoted by R. The list, $R[1], \ldots, R[m], 1 \leq m \leq F$, is updated during the algorithm, that is, we remove from R requests that were fulfilled, and we move to their updated place requests that were only partially fulfilled. The bakeries are given in a non-decreasing order of their storage capacity, i.e., $C_1 \leq C_2 \leq, \ldots, \leq C_N$.

The algorithm uses two bakery-filling procedures: some of the bakeries are filled using the *greedy-filling* procedure, while for other bakeries the *moving-window* procedure is applied. First we describe the two procedures and then we present the algorithm.

Greedy-Filling:
The greedy-filling procedure allocates bagels from a bakery to the requests, starting from the smallest request, $R[1]$, and continuing until the bakery is *saturated*, that is, until it allocates exactly Q_j bagels. The last request may split. A formal description of the greedy-filling procedure is given in Figure 2.

Moving-Window:
The moving-window procedure fills a bakery, b_j, with the first sequence of C_j requests that require together at least Q_j bagels. We search the request list R using a moving window of size C_j. At first, the window covers the set of the smallest C_j requests. In every iteration we replace the smallest element in the window by the next element in R, until the requests that are included in the window are large enough to saturate b_j. If even the subset of the largest C_j requests requires less than Q_j bagels, then we can not saturate b_j. However, we show below that this never happens. A formal description of the moving-window procedure is given in Figure 3. Note that the window advances until it covers C_j requests such that $Q_j > R[i-1] + \ldots + R[i+C_j-2]$ and $Q_j \leq R[i] + \ldots + R[i+C_j-1]$. At this stage we can clearly bake in b_j all the demand of the $C_j - 1$ flavors $R[i], \ldots, R[i+C_j-2]$ and saturate b_j by baking some of the bagels required of the flavor $R[i+C_j-1]$.

The Algorithm $\mathcal{A}_u$:
The algorithm proceeds in iterations, in the jth iteration we fill b_j according to the following rules:

1. If there are less than C_j requests or if the subset of the smallest C_j requests requires more than Q_j bagels, then fill b_j using the greedy procedure.
2. If the subset of the smallest C_j requests requires at most Q_j bagels, then fill b_j using the moving-window procedure.

In order to show that the algorithm terminates with a perfect assignment we distinguish between two stages in the execution of $\mathcal{A}_u$:

1. As long as each bakery fulfills $C_j - 1$ or C_j requests. This stage includes executions of the moving-window procedure and some executions of the greedy procedure.
2. Starting at the first time some bakery fulfills less than $C_j - 1$ requests. This can clearly happen only when the greedy procedure is applied.

Note, that for some inputs, one of the stages may be empty. For each of the two stages, we show that each of the bakeries that are filled during this stage allocates exactly Q_j bagels to at most C_j flavors. The proofs of the following lemmas are omitted.

Lemma 3.4 *Each of the bakeries that are filled during the first stage will produce exactly Q_j bagels of at most C_j flavors.*

Lemma 3.5 *Each of the bakeries that are filled during the second stage will produce exactly Q_j bagels of at most C_j flavors.*

By combining Lemmas 3.4 and 3.5 we get that each of the bakeries will produce exactly Q_j bagels of at most C_j flavors. Thus, the algorithm terminates with a perfect assignment.

The algorithm is polynomial: Each of the filling procedures takes $O(F)$ steps. Adding the complexity of sorting the lists, the total complexity of the algorithm is $O(N \cdot F + N \log N + F \log F)$.

■

3.3.1 Approximating Uniform Capacity Ratio

"That's really nice of them", admired Noah, "I didn't assume Bigbird can do better than counting to ten". Noah called Jeppeto and asked him to distribute cabinets among the bakeries in a way that achieves uniform capacity ratio. Formally, it means that Noah turned to deal with a slightly different model where each bakery has its baking capacity, Q_j, and in addition, there is a limit, C, on the *total* number of cabinets that can be installed in the bakeries. Jeppeto should determine now the number of cabinets, C_j, in each bakery, such that $\sum_{j=1}^{N} C_j = C$.

"All I need now is to determine $C_j = \frac{Q_j}{Q} \cdot C$", said Jeppeto, "This way I achieve uniform capacity ratio with $r = \frac{Q}{C}$". Jeppeto started the calculations and he soon realized that things were not that simple. The number of cabinets in each bakery must be an integer. "I must round the C_j's in a way that will minimize the violation of uniformity". He thought and formulated his problem as the following integer programming problem:

$$\text{Minimize} \quad \frac{\max \frac{Q_j}{C_j}}{\min \frac{Q_j}{C_j}} \quad \text{such that} \quad \sum_{j=1}^{N} C_j = C, \quad C_j \text{ integer.}$$

Looking at the 'Fair Resource Allocation' chapter in [5], Jeppeto found out that problems of this type can be optimally solved in $O(N \log N + N \log C)$ steps. It provides an allocation of the cabinets to the bakeries in a way that minimizes α such that $\forall j,\ r \leq \frac{Q_j}{C_j} \leq \alpha \cdot r$. "It looks that I can only approximate the perfect assignment", Jeppeto concluded. He then wrote the following theorem:

Theorem 3.6 *Let α be the minimal number such that there exists r satisfying $\forall j,\ r \leq \frac{Q_j}{C_j} \leq \alpha \cdot r$. Then an assignment with revenue $\frac{1}{\alpha} \cdot Q$ can be found in polynomial time.*

Proof: For each bakery b_j, determine $Q'_j = \lceil r \cdot C_j \rceil$. Let $Q' = \sum_{j=1}^{N} Q'_j$. For each flavor f_i, determine $W'_i \leq W_i$ such that $\forall i, W'_i$ is an integer, and $\sum_{i=1}^{F} W'_i = Q'$ We obtain a system with nearly uniform capacity ratio. It can be seen that the algorithma $\mathcal{A}_u$ presented in Section 3.3 is suitable for this system. The perfect assignment for the new system is legal and has revenue at least $\frac{1}{\alpha} \cdot Q$. ■

"In my case", thought Jeppeto, "it can be seen that $\alpha \leq \frac{C_k}{C_k - 1}$ where b_k is the bakery with the maximal capacity ratio".

Corollary 3.7 *If cabinets can be moved, an assignment with revenue $\frac{C_k - 1}{C_k} Q$ can be found in $O(N \cdot F + N \log N + N \log C)$ steps, where b_k is the bakery in which the resulting capacity ratio is maximal.*

"However, I can't adopt these results" claimed Jeppeto "Some of the bakeries have large baking capacities and small variety bounds. The idea of moving cabinets is nice but it is not applicable, there is no room for additional cabinets in some of the bakeries. Also some of the cabinets are attached to the wall and can't be removed. I think I'll put my efforts in approximating the optimal assignment assuming cabinets can not be moved".

4 Approximating the Optimal Assignment

4.1 Increasing the Storage Capacity

Recall, that Jeppeto's suggestion to Noah was to add a *single* cabinet to each bakery. Noah despised this idea, but Jeppeto didn't give up and proved that it is enough to increase by one the storage capacity of each bakery, in order to eliminate the gap between the performance of the optimal, probably exponential, algorithm, and a polynomial algorithm.

Theorem 4.1 *If the storage capacity of each bakery is increased by one, then the revenue problem can be solved optimally in polynomial time.*

Proof: We present a polynomial time algorithm that terminates with an optimal assignment. The algorithm, denoted by $\mathcal{A}_r$, proceeds bakery after bakery, for each bakery b_j, it decides on the production of at most Q_j bagels made of at most $C_j + 1$ flavors. The total number of bagels produced according to $\mathcal{A}_r$ is at least the total number of bagels produced according to an optimal algorithm that produces in b_j at most Q_j bagels of at most C_j flavors.

As in the uniform-ratio case (Section 3.3), we assume that the flavors are given in a non-decreasing order of their popularities, i.e. the requests satisfy $W_1 \leq W_2 \leq, \ldots, \leq W_F$. We keep the remaining requests in a sorted list, denoted by R, that is updated during the algorithm. This algorithm also uses the two bakery-filling procedures: *greedy-filling* and *moving-window* introduced in Section 3.3. However, the moving-window procedure is changed as follows: First, the window covers $C_j + 1$ requests (instead of C_j). Second, if the subset of the largest $C_j + 1$ requests requires less than Q_j bagels, then we can not saturate b_j and we do the best we can: we fill b_j with these $C_j + 1$ requests (In the uniform-ratio case this never happens).

The bakeries are given in a non-increasing order by their *capacity ratio*, i.e., $\frac{Q_1}{C_1} \geq, \ldots, \geq \frac{Q_N}{C_N}$. The sorted bakeries are kept in a list denoted by B'. Another list of bakeries, denoted by B'', may be created during the execution of the algorithm.

The Algorithm $\mathcal{A}_r$:

Generally, the algorithm uses the moving-window procedure to fill the bakeries according to their order in B', that is, in the jth iteration we fill b_j and remove it from B'.

Note, that the moving-window procedure allocates exactly $C_j + 1$ flavors to b_j. Sometimes, it is not possible to allocate that amount of flavors. In such cases we move the bakery to B'' or look for another bakery for which the moving-window procedure may be applied. Bakeries that are moved from B' to B'' are filled using the greedy procedure after B' is empty.

The following rules are used when the next bakery examined by $\mathcal{A}_r$ is $b_j \in B'$.

- If $b_j = \emptyset$, i.e., B' is empty, fill sequentially all the bakeries in B'' using the greedy-filling procedure.
- If there are less than $C_j + 1$ requests in R, move b_j to the end of B''.
- If the subset of the smallest C_j requests in R requires at most Q_j bagels, fill b_j with $C_j + 1$ requests according to the moving-window procedure.
- If the subset of the smallest C_j requests requires more than Q_j bagels, look for the first bakery $b_k \in B'$ for which the subset of the smallest C_k requests requires at most Q_k bagels. If such a bakery exists, fill it according to the moving-window procedure. If there is no such bakery in B', fill all the bakeries in B' and B'' using the greedy-filling procedure.

Optimality: Let us denote by $G = \{b_{g_1}, b_{g_2}, \ldots, b_{g_m}\}$ the set of bakeries that are *not saturated* by $\mathcal{A}_r$, and by $z_1, z_2, \ldots, z_m$ the resulting waste of bagels on each non-saturated bakery, i.e., b_{g_i} produces only $Q_{g_i} - z_i$ bagels. We show that there is no legal assignment in which the total revenue exceeds $\sum_{j=1}^{N} Q_j - (z_1 + z_2 + \ldots + z_m)$. We distinguish between two stages of $\mathcal{A}_r$:

1. Bakeries from B' are filled using the moving-window procedure. Some bakeries may be moved to B''.

2. Bakeries from B' and B'' are filled using the greedy-filling procedure, (when no bakery from B' can be filled by the moving-window procedure and later, when B' is empty).

For the first stage we show that the total waste of bagels (non-used baking capacity) does not exceed the waste in the optimal assignment. For the second stage we show that there is no waste at all, that is, all the bakeries filled during this stage are saturated.

For simplicity, assume that whenever the moving-window-procedure is executed, the list R is scanned from left to right, that is, the smallest request, $R[1]$, is the leftmost request and the largest request is the rightmost one. During the execution of the moving-window procedure, the window moves from left to right.

We can consider the requests fulfilled by b_j as a *hole* that is created in R. Let us examine the sequence of holes created in R during the first stage of the algorithm. We show that every non-saturated bakery creates at the right end of R a hole that is the union of all the holes in R.

Claim 4.2 *Every $b_{g_j} \in G$ filled during the first stage of $\mathcal{A}_r$ unites the holes existing in R into a single hole that is positioned at the right end of R.*

Proof: The proof is by induction on j, the index of b_{g_j} in G. We omit the details. ∎

From the way the holes are created we conclude that $\mathcal{A}_r$ is optimal for the bakeries filled during the first stage. The proof is omitted.

Lemma 4.3 *Let b_j be the last bakery filled by $\mathcal{A}_r$ during the first stage, then the total number of bagels produced in $b_1, \ldots, b_j$ is at least the number of bagels produced in $b_1, \ldots, b_j$ by an optimal algorithm.*

In order to complete the optimality proof of the algorithm we show that all the bakeries that are filled during the second stage of $\mathcal{A}_r$ are saturated. The proof is omitted.

Lemma 4.4 *Each of the bakeries that are filled during the second stages produces exactly Q_j bagels made of at most $C_j + 1$ flavors.*

By combining Lemmas 4.3 and 4.4 we get that total amount of wasted bagels do not exceed the total amount of wasted bagels in the optimal assignment. In particular, if there exists a perfect assignment on a system, then our algorithm finds a perfect assignment in a system in which the storage capacities are increased by one.

The algorithm is polynomial: Each of the filling procedures has complexity $O(F)$. In the worst case it takes $O(F \cdot N)$ steps to choose the next bakery to be filled. Therefore, the total complexity of $\mathcal{A}_r$ is $O(F^2 \cdot N^2)$. ∎

Acknowledgments

We are indebted to Noah for his devoted work on this problem. In turn, Noah would like to thank Jeppeto for his contribution to the approximation results, and to his friends from Sesame St. for their helpful comments.

References

[1] *Best of Berkeley: Best Bagels*, http://www.dailycal.org/bestof/headlines/bagels.html

[2] D.K. Friesen and M.A.Langston, *Variable Sized Bin Packing*, SIAM J. Compute. Vol 15, No. 1, February 1986, pp. 222-230.

[3] M.R. Garey and D.S. Johnson, *Computers and intractability: A Guide to the Theory of NP-Completeness*, W.H. Freeman, (1979).

[4] R.L. Graham, *Bounds on multiprocessing timing anomalies*, SIAM J. Appl. Math., 17:263-269, 1969.

[5] T. Ibaraki and N. Katoh, *Resource Allocation Problems - Algorithmic Approaches*, The MIT Press, 1988.

[6] K.L. Krause, V.Y. Shen, and H.D. Schwetman, *Analysis of Several Task-Scheduling Algorithms for a model of Multiprogramming Computer Systems*, Journal of the ACM, Vol 22, No. 4, October 1975, pp. 522-550.

[7] J.L. Wolf, P.S. Yu, and H. Shachnai, *Disk Load Balancing for Video-on-Demand Systems*, ACM Multimedia Systems Journal, Vol. 5, 1997, pp. 358-370.

Appendix A

Proof of Theorem 3.1: We represent the system by a bipartite graph $G = (V_1, V_2, E)$, where V_1 consists of F nodes corresponding to the flavors, and V_2 consists of N nodes corresponding to the bakeries. There are edges connecting each flavor to all the bakeries (a complete bipartite graph). Consider a flow network based on G with one source and one sink. The source, s, is a new node connected to all the flavors. The capacity of each edge $\langle s, f_i \rangle$ is equal to W_i. The sink, t, is a new node connected to all the bakeries. The capacity of each edge $\langle b_j, t \rangle$ is equal to Q_j. Any edge $\langle f_i, b_j \rangle$ has capacity ∞. Apply a polynomial time algorithm for maximum flow on this network. The assignment induced by the flow is:

- $A_{i,j} = 1$ iff the flow on the edge $\langle f_i, b_j \rangle$ is not 0.
- $B_{i,j} = k$ iff the flow on the edge $\langle f_i, b_j \rangle$ is k.

The assignment is legal:

- Clearly, $A_{i,j} = 0 \Rightarrow B_{i,j} = 0$.
- For each bakery b_j, $\sum_i B_{i,j} \leq Q_j$, because the flow is legal, and Q_j is the flow capacity of $\langle b_j, t \rangle$.
- For each bakery b_j, $\sum_i A_{i,j} \leq C_j$: Since for all the bakeries $Q_j \leq C_j$ and the minimal non-zero flow is 1.

The assignment is perfect: Since $\sum_i W_i = Q = \sum_j Q_j$, the algorithm terminates with the two cuts (s, V_1) and (V_2, t) saturated. This means that for every i, exactly W_i bagels are allocated for the flavor f_i. ■

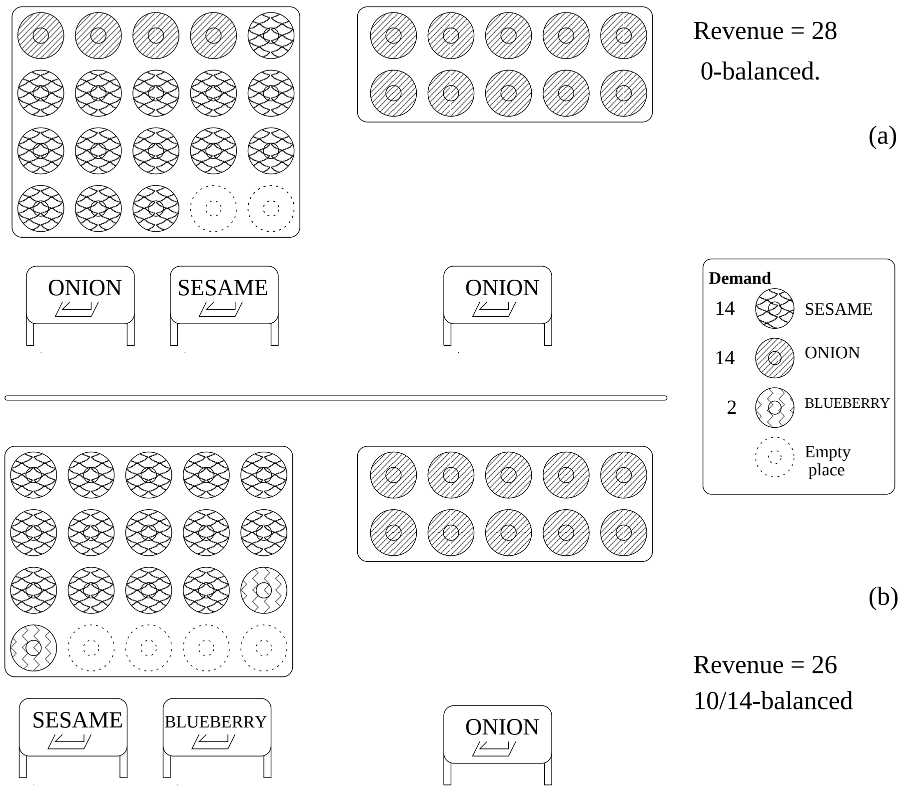

Figure 1: Maximal Revenue vs. Maximal Balance

Appendix B: Application to MOD Systems

In this section we show how Noah's results can be used for the efficient allocation of resources in multimedia computer systems. As noted in section 1.1, the revenue problem refers to the allocation of video program files to disks such that the total broadcast potential of the system is maximized. The storage capacity is measured as the number of files that can be stored on each of the disks. By Theorem 4.1 we have:

Theorem 4.5 *If we increase by one the number of files that can be stored on each of the disks, then the revenue problem can be solved optimally in polynomial time.*

By Corollary 3.7 the utilization of the system can be maximized if the storage resources are distributed among the disks in a way that maximize the capacity ratio uniformity. Formally:

Theorem 4.6 *If storage resources can be distributed among the disks, an assignment with revenue* $\frac{C_k - 1}{C_k} Q$ *can be found in* $O(N \cdot F + N \log N + N \log C)$ *steps, where* k *is the disk with the largest resulting capacity ratio.*

In particular, a perfect assignment can be found if the capacity ratio is uniform. The other conditions for perfect assignment presented in Section 3 are also applicable to MOD systems:

Theorem 4.7 *If the storage capacity of each disk is not smaller than its broadcast capacity, or if for all the disks* $C_j \geq \frac{Q_j \cdot F}{\varepsilon \cdot Q} + 1$ *where* ε *is the maximal number such that for all the files* $p_i \geq \frac{\varepsilon}{F}$*, then a perfect assignment exists and can be found in polynomial time.*

Greedy-Filling(j)
$i \leftarrow 1$
repeat
  Assign in b_j a cabinet for the flavor f_k corresponding to $R[i]$.
  Produce in b_j bagels of the flavor f_k: $B_{k,j} = \min\{R[i], Q_j\}$
  $Q_j \leftarrow Q_j - B_{k,j}$
  $i \leftarrow i + 1$
until $Q_j = 0$
Remove from R the requests that were fulfilled by b_j.
Update the value of the new smallest request that was only
  partially fulfilled by b_j.
Remove b_j from the bakery list.

Figure 2: The *Greedy-Filling* Procedure

Moving-Window(j)
$i \leftarrow 1$
$C = R[1] + \ldots + R[C_j]$
while $(C < Q_j$ and $|R| \geq C_j + i)$
  $C \leftarrow C - R[i] + R[C_j + i]$
  $i \leftarrow i + 1$
Fill b_j:
  Assign in b_j cabinets for the flavors corresponding to the requests
    $R[i], \ldots, R[i + C_j - 1]$.
  For each k, $i \leq k \leq C_j + i - 2$:
    Produce in b_j $R[k]$ bagels of the corresponding flavor.
  Allocate some bagels to the flavor $f_{k'}$ corresponding to $R[i + C_j - 1]$:
  $B_{k',j} = Q_j - (R[i] + \ldots + R[i + C_j - 2])$.
Update the list R:
  Remove from R the requests $R[i], \ldots, R[i + C_j - 2]$
  if the requirement of f_k, the flavor corresponds to $R[i + C_j - 1]$, is not fulfilled
   Move $R[i + C_j - 1]$ to its appropriate new place in the sorted list R.
Remove b_j from the bakery list.

Figure 3: The *Moving-Window* Procedure