RC23721 (W0509-015) September 7, 2005
Mathematics

IBM Research Report

The Optimality of the On-Line Greedy Algorithm in Carpool and Chairman Assignment Problems

Don Coppersmith, Tomasz J. Nowicki, Giuseppe A. Paleologo, Charles Tresser, Chai Wah Wu
IBM Research Division
Thomas J. Watson Research Center
P.O. Box 218
Yorktown Heights, NY 10598

THE OPTIMALITY OF THE ON-LINE GREEDY ALGORITHM IN CARPOOL AND CHAIRMAN ASSIGNMENT PROBLEMS

ABSTRACT. We study several classes of related scheduling problems including the carpool problem, its generalization to arbitrary inputs and the chairman assignment problem. We derive both lower and upper bounds for on-line algorithms solving these problems. We show that the greedy algorithm is optimal among on-line algorithms for the chairman assignment problem and the generalized carpool problem. We also consider geometric versions of these problems and show how the bounds adapt to these cases.

1. INTRODUCTION

In this paper, we *mostly* consider the following setting for a class of scheduling problems: Let Δ denotes the standard $n-1$ dimensional simplex in the real affine space $\mathcal{A}^n$ which for all practical reasons can be identified with $\mathbb{R}^n$:

$$\Delta = \left\{ \mathbf{\Lambda} = (\lambda_1, \lambda_2, \ldots, \lambda_n) \in \mathcal{A}^n : \sum_{i=1}^{n} \lambda_i = 1 \text{ and } \lambda_i \geq 0 \right\} .$$

As the case $n = 1$ is trivial, we will always assume $n \geq 2$.

- There are n *tasks* $\mathbb{T}_j$, $j = 1, \ldots, n$, each one represented by one corner (or vertex) of the simplex Δ: $\mathbf{c}_j \in \mathcal{C} = \{\mathbf{c}_i, i = 1, \ldots, n\}$.
- At each (discrete) moment of time $t \in \mathbb{N}^* = \{1, 2, \ldots\}$, a *demand* is requested, such demand (or *input*) being represented as the point $\mathbf{\Lambda}(t) \in \Delta \subset \mathcal{A}^n$.
- Then the *task at time t* is chosen $\mathbb{T}(t) = \mathbb{T}_{j(t)}$, and the corresponding corner of Δ is considered as the *output* $\mathbf{c}(t) = \mathbf{c}_{j(t)}$ at time t, of a
- process which is determined:
  – by the sequence of inputs, and
  – by the method chosen

  to get a *schedule*, *i.e.,* an output sequence $\left(\mathbf{c}_{j(t)}\right)_{t\in\mathbb{N}^*} = \mathbf{c}_{j(1)}, \mathbf{c}_{j(2)}, \ldots$ out of an input sequence $(\mathbf{\Lambda}(t))_{t\in\mathbb{N}^*} = \mathbf{\Lambda}(1), \mathbf{\Lambda}(2), \ldots$, a method that we call for now an
- *algorithm* $\mathbb{A}$ (a concept to be formally defined below in a way appropriate for our purpose).
- The collections $\mathcal{I}$ of possible inputs and $\mathcal{O}$ of admissible choices of outputs define for us the *problem class* $\Pi(\mathcal{I}, \mathcal{O})$. We often say *problem* for problem class, especially when dealing with problem classes that bear special names; we hope that this will not cause any confusion since then, the usual meaning of the word problem would work as well.
- A problem class together with restrictions on the algorithms that are considered as acceptable define a *protocol* $\mathcal{P}$ of a scheduling problem (all algorithms may be acceptable, yielding the most general protocol for a given problem class).

Remark. *The simplex Δ is more often understood as a subset of the n-dimensional real affine space than as a subset of the corresponding real vector space since both inputs and outputs are naturally modeled by points (hence elements of affine spaces rather than vectors), while objects such as errors get modeled by vectors since they appear as differences of points (something which appeals even more for the point space to be an affine space).*

We notice also that in some cases related to the ones that we consider here, outputs may be collections of tasks or some number (the weight*) of copies of a single task , so that both inputs and outputs may be more generally modeled as collections of weighted points or as the points from a convex combinations of the corners and the corners themselves of some polytopes in (generally) lower dimensional affine space.*

The word "mostly" used in the first sentence of the paper refers to the fact that a more general situation will also be considered where multiple simplices represent the inputs and their vertices the outputs. This more general setting appears naturally in the geometrical formulation of problems classes that have inputs in a single simplex, but which have special protocols with constraints such as:

"If the input belongs to an m-face of the simplex,
then the output belongs to the same m-face."

In brief, we will study some algorithms for *scheduling problems* that are represented geometrically as above. In order to state more precisely the problems that we will consider, we need next to introduce more quantities, concepts, and notations associated with the inputs and the outputs.

Comparing the inputs and the outputs gives rise to the time depending sequence of *cumulative error vectors* defined by:

$$\mathbf{E}(t) = \sum_{s=1}^{t} (\mathbf{\Lambda}(s) - \mathbf{c}(s)) \in \mathbb{R}^n \,. \tag{1.1}$$

Remark that as both $\mathbf{\Lambda}(s)$ and $\mathbf{c}(s)$ belong to Δ then:

$$\sum_j E_j(s) = 0, \qquad \text{where} \qquad \mathbf{E}(s) = (E_1(s), \ldots, E_n(s))\,. \tag{1.2}$$

We only consider here the case when $\mathbf{E}(0) = (0, \ldots, 0)$ since we are interested in actual bounds, while studies such as [1] or [10] on the existence of bounds rather consider more general initial error vectors. Note that the error can be written recursively as:

$$\mathbf{E}(t) = \mathbf{E}(t-1) + \mathbf{\Lambda}(t) - \mathbf{c}(t)\,; \quad t \in \mathbb{N}^*\,, \tag{1.3}$$

and as we always set $\mathbf{E}(0) = 0$ we have again (1.2). Notice that while inputs and outputs are defined for $t \in \mathbb{N}^*$, errors are defined for $t \in \mathbb{N} = 0, 1, 2, \ldots$. Now we introduce the *modified input* $\mathbf{\Theta}(t) = \mathbf{E}(t-1) + \mathbf{\Lambda}(t)$. Using the formula (1.1) this can be rewritten as:

$$\mathbf{\Theta}(t) = \mathbf{\Theta}(t-1) + \mathbf{\Lambda}(t) - \mathbf{c}(t-1)\,; \quad t \in \mathbb{N}^*\,, \tag{1.4}$$

which is a time dependent dynamical system in the affine space $\mathcal{A}^n$. More precisely, because of (1.2) the dynamical system defined by (1.4) acts in the hyperplane $\sum_j x_j = 1$ (notice that $\mathbf{c}(t-1)$ only depends on $\mathbf{\Theta}(t-1)$ and on the chosen algorithm $\mathbb{A}$). With this notion the representation of $E_j(t)$ can be written as [1]:

$$E_j(t) = \begin{cases} \Theta_j(t) - 1 & = E_j(t-1) + \lambda_j(t) - 1 \quad \text{if } j = j(t) \\ \Theta_j(t) & = E_j(t-1) + \lambda_j(t) \quad \text{otherwise} \end{cases} . \tag{1.5}$$

where $\Theta_j(t) = E_j(t-1) + \lambda_j(t)$ by definition of $\mathbf{\Theta}(t)$.

Remark. *There is an alternate time dependent dynamical system* (1.3)*, this time in the vector space* $\mathbb{R}^{(n-1)}$ *since errors are vectors, and we have* (1.2)*. We call* $E_j(t)$ *the* cumulative error of task $\mathbb{T}_j$ at time t *and we omit the argument* t *when the time is clear from the context. Note that again by* (1.2) *the modified input can be written as* $\mathbf{\Theta}(t) = \sum_j \Theta_j(t)\mathbf{c}_j$ *with* $\sum_j \Theta_j(t) = 1$.

More generally, without (1.2)*,* $\sum_{j=1}^n E_j(t)$ *is a constant as* t *varies. However, with* (1.2)*, all modified inputs belong to the hyperplane that contains the standard simplex, which means that the dynamics of the modified inputs takes place in the* $(n-1)$*-dimensional affine subspace, a point of view used elsewhere to consider inputs in more general polytopes (in which case, the results one can expect are of course less precise than what will be described here): see for instance* [1], [2], [3], [10].

Having introduced the notions of the input, the output and the error we are ready to talk about the algorithms.

- Abstractly an *algorithm* will be for us a function $\mathbb{A}$ which maps the space $\{(\mathbf{\Lambda}(t))_t\}$ of infinite demand sequences $(\mathbf{\Lambda}(t))_{t\in\mathbb{N}^*} \in \Delta^{\mathbb{N}^*}$ into the space $\{(\mathbf{c}(t))_t\}$ of infinite outputs sequences $(\mathbf{c}(t))_{t\in\mathbb{N}^*}$, and further into the space $(\mathbb{T}(t))_t \in \{1,\ldots,n\}^{\mathbb{N}^*}$ of infinite task sequences:
$$\mathbb{A} : (\mathbf{\Lambda}(t))_{t\in\mathbb{N}^*} \mapsto (\mathbb{T}(t))_{t\in\mathbb{N}^*} .$$
Notice that this definition of an algorithm is not a standard one [2].
- More generally given sets of inputs $\mathcal{I}$ and outputs $\mathcal{O}$ the protocol is defined by a subset of the set of maps $(\mathcal{O}^{\mathbb{N}})^{(\mathcal{I}^{\mathbb{N}})}$ and an algorithm is any map from this subset.
- The properties, conditions, or restrictions attached to a protocol define the *context* of the problem, which is to find an algorithm not only compatible with a given protocol, but also fulfilling some additional qualitative or quantitative conditions, such as providing best value of the some bounds, or limiting the information that can be used such as requiring the algorithm to be on-line.
- Usually such conditions are given by a collection of inequalities and at some points the ambiguities have to be resolved by tie breaking rules, that are either deterministic or probabilistic, and have to be included in algorithms.

In this paper the main issues that we will tackle for the protocols that we consider will be:

[1]When the $\mathbf{c}_j$'s are not linearly independent, the decomposition $\mathbf{E}(t) = \sum_{j=1}^n E_j(t)\mathbf{c}_j$ is not unique. On the other hand, for several of the scheduling problems that we consider, we set the $\mathbf{c}_j$'s to be such that the vectors $\mathbf{c}_j - 0$ are unit coordinate vectors, in which case $E_j(t)$ is unique.

[2]See the fundamental book by Donald E. Knuth [7], where the algorithm is a map from a space of states to itself, such that given a subsets of states called inputs and a subsets of states called outputs we impose that (a) every input is mapped in a finite number of steps to an output and (b) every output is a fixed point of the map. The finiteness may be not uniform and is not required for a broader notion of *computational method*.

- to find the best algorithm (under some protocol or further constraints) and/or
- to estimate how good or bad an algorithm can be.
- The *quality of an algorithm* will be measured by the supremum over time and over all possible input sequences (under some protocol) of the norms of $\mathbf{E}(t)$.

For scheduling problems, this norm is usually the sup norm $\|\mathbf{E}(t)\|_\infty$ considered for each time t in the appropriate space. To this effect for each algorithm $\mathbb{A}$ and demand sequence $\{\boldsymbol{\Lambda}(t)\}_t$ acceptable by some protocol $\mathcal{P}$ we define the *bound relative to the input sequence* as:

$$B_{\mathbb{A}}(\{\boldsymbol{\Lambda}(t)\}_t) = \sup_{t\in\mathbb{N}} \|\mathbf{E}(t)\| \ , \tag{1.6}$$

from which the *universal bound for the algorithm* $\mathbb{A}$ is obtained as:

$$\mathbf{B}_{\mathbb{A}} = \sup_{\{\{\boldsymbol{\Lambda}(t)\}_t\}_{\boldsymbol{\Lambda}}\in\mathcal{P}} \left(B_{\mathbb{A}}(\{\boldsymbol{\Lambda}(t)\}_t)\right), \tag{1.7}$$

and then the *absolute bound for the protocol at hand* as:

$$\mathbf{B} = \inf_{\mathbb{A}\in\mathcal{P}} \mathbf{B}_{\mathbb{A}} \, . \tag{1.8}$$

We will be interested in *on-line algorithms*, *i.e.,* algorithms that can be represented as sequences of compatible maps from initial segments of inputs to initial segments of outputs of the same length:

$$\mathbb{A}_t : (\boldsymbol{\Lambda}(s))_{s\le t\in\mathbb{N}} \mapsto (\mathbb{T}(s))_{s\le t\in\mathbb{N}} \, .$$

Otherwise speaking, on-line means here that the choice of $\mathbb{T}(t) = \mathbf{c}(t)$ depends only on $\boldsymbol{\Lambda}(s)$, for $s \le t$ (and hence can also depend on $\mathbb{T}_{j(s)}$ for $s < t$ once the algorithm has been chosen, as a compact or otherwise convenient way to represent the dependance on past and present inputs).

We will give bounds for $B_{\mathbb{A}}$ for several on-line algorithms. The lower bound will be given for all on-line algorithms, whereas the upper bound will be for a specific on-line algorithm. For the precise formulation see Definition 1.

In several cases the specific algorithm which gives the best upper bound is the *greedy algorithm*, the algorithm that at each time chooses $\mathbb{T}(t) = \mathbb{T}_{j(t)}$ among the allowable tasks such that the distance between the modified input $\boldsymbol{\Theta}(t) = \mathbf{E}(t-1) + \boldsymbol{\Lambda}(t)$ and $\mathbf{c}_{j(t)}$ is the smallest possible, with a deterministic or random tie breaker when several $\mathbf{c}_j$'s are at the same distance from the modified input. We borrow this terminology from *error diffusion*, a greedy algorithm that is sometimes used to solve the problem of halftoning that one faces in order to print digital images with a digital printer [2].

The i^{th} coordinate of a demand is naturally associated to the corner $\mathbf{c}_i \in \mathcal{C}$, and so is the i^{th} coordinate of a modified input: the correspondence goes both way so that one can speak without ambiguity, *e.g.,* of the *input of a task or of a corner* or of the *modified input to a task or to a corner (then taken as representing the coordinate axis that contains it)* (see for instance the formulation of Proposition 2.5).

We consider the following classes of algorithms in order of increasing constraints on the chosen task $\mathbb{T}_{j(t)}$ and on the demand $\boldsymbol{\Lambda}(t)$. In sections 3-6 we assume $\mathbf{c}_j$ to be unit coordinate vectors $\mathbf{e}_j$, in which case $\|\mathbf{E}(t)\|_\infty = \max_j E_j(t)$ and

$B_{\mathbb{A}}(\mathbf{\Lambda}) = \sup_t \max_j |E_j(t)|$. Furthermore, in this case the greedy algorithm picks the task $\mathbb{T}_j$ among all allowable tasks such that $\Theta_j(t)$ is the largest. In Sections 7-8 we consider more general $\mathbf{c}_j$'s. The paper is organized as follows, where for each section from 3 to 8 we indicate what is *the set of inputs* $\mathcal{I}$ (which is Δ in most but not all cases) and *the set of outputs* $\mathcal{O}$ (which is always the set of corners of the set of inputs):

- In Section 2, we review **General properties of on-line algorithms** and related issues.
- In Section 3, we consider **The Chairman Assignment Problem (CAP)** [12] where $\mathcal{I} = \Delta$, $\mathcal{O} = \{\mathbf{e}_j\}$, and there is no restriction on the demands and tasks.
- In Section 4, we consider **The Generalized Carpool Problem (GCP)** for which $\mathcal{I} = \Delta$, $\mathcal{O} = \{\mathbf{e}_j\}$, and a task can be chosen as output only if the corresponding demand entry is not zero. This means that the driver of the carpool must be in the car, *i.e.,* in symbols: $\lambda_{j(t)} > 0$.
- In Section 5, we revisit the **Carpool Problem (CP)** [6] to which one of us (D.C.) has already made contributions in the past (see [6] for details). The Carpool Problem is the same as the Generalized Carpool Problem except that the nonzero entries of a demand are all equal to each other (hence equal to one divided by the number of non-zero entries).
- In Section 6, we study **The Chairman Assignment Problem with Multiple Tasks (MCAP)** where for a $\sigma = \frac{m}{d} \in \mathbb{Q}$ the set $\mathcal{I}$ of inputs is $\left\{p : \sum_j p_j = \sigma,\, p_j \in [0,1]\right\}$ and the set $\mathcal{O}$ of outputs form the set, say $\mathcal{O}_M CAP$, of *all the rational points of the polytope* $\mathcal{I}$ *with the denominator* d.
- In Section 7, we discuss the **Geometric Version** of the problems that we consider here.
- In Section 8, we consider **The Multiple Polytope Problem** for which $\mathcal{I} = \prod P_i$, $\mathcal{O} = \prod \mathcal{O}_i$ with $\mathcal{O}_i$ being the corners of the polytope P_i. At each time $t \in \mathbb{N}$, first a polytope is given out of some collection $\{P_i\}$, then a demand represented by a point from this polytope P_j is given.The task must be chosen as a corner of the same polytope P_j. See Section 8 for the formulation which fits the general setting.

Remark. *Except otherwise stated, all algorithms are online: we consider more general algorithms at the beginning of Section 2.*

2. General properties of on-line algorithms

We are going first to revisit the quantities introduced in equations (1.6), (1.7), and (1.8) in the context of general protocols and protocols that are assuming on-line algorithms.

Definition 1 (**Lower bound, upper bound**). *We say that the number L is a* lower bound *for a protocol $\mathcal{P}$ (respectively a protocol $\mathcal{P}$ assuming on-line algorithms) if for every on-line algorithm $\mathbb{A} \in \mathcal{P}$ there is a demand sequence $\mathbf{\Lambda}_{\mathbb{A}} \in \mathcal{I} \in \mathcal{P}$ such that $B_{\mathbb{A}}(\mathbf{\Lambda}_{\mathbb{A}}) \geq L$.*

We say that the number U is an upper bound *if there exists an algorithm $\mathbb{A}_0$, for which every demand sequence $\mathbf{\Lambda}$ yields $B_{\mathbb{A}_0}(\mathbf{\Lambda}) \leq U$.*

Clearly, by an upper bound we mean here an acceptable upper bound in the sense that any algorithm that does not respect this bound is unsatisfactory, but the bound can be enforced. We have the following Proposition linking the concepts introduced in Definition 1.

Proposition 2.1 (**Tight bound**). *Let:*

$$\begin{aligned} \mathbf{B}_{\mathbb{A}} &= \sup_{\mathbf{\Lambda}} B_{\mathbb{A}}(\mathbf{\Lambda}) \\ \mathbf{B} &= \inf_{\mathbb{A}} \mathbf{B}_{\mathbb{A}} \,. \end{aligned}$$

Then for any lower bound L and any upper bound U:

$$L \leq \mathbf{B} \leq U \,.$$

Moreover:

$$\mathbf{B} = \sup L = \inf U \,,$$

where the supremum is taken over all lower bounds L and the infimum is taken over all upper bounds U.

Proof. If L is a lower bound then for every algorithm $\mathbf{B}_{\mathbb{A}} \geq L$, hence $\mathbf{B} \geq L$. If U is an upper bound then for some $\mathbb{A}_0$, $\mathbf{B}_{\mathbb{A}_0} \leq U$, hence $\mathbf{B} \leq U$. In particular $\sup L \leq \mathbf{B} \leq \inf U$.

By definition of the supremum, for every $\epsilon > 0$ and every $\mathbb{A}$ there exists a demand $\mathbf{\Lambda}_{\epsilon,\mathbb{A}}$ such that $B_{\mathbb{A}}(\mathbf{\Lambda}_{\epsilon,\mathbb{A}}) \geq \mathbf{B}_{\mathbb{A}} - \epsilon \geq \mathbf{B} - \epsilon$, as $\mathbf{B} = \inf_{\mathbb{A}} \mathbf{B}_{\mathbb{A}}$. Hence $\mathbf{B} - \epsilon$ is a lower bound. We proved $\sup L \geq \mathbf{B}$.

On the other hand for every $\mathbf{\Lambda}$ we have $B_{\mathbb{A}}(\mathbf{\Lambda}) \leq \mathbf{B}_{\mathbb{A}}$. Then by definition of infimum for every $\epsilon > 0$ there exists an algorithm $\mathbb{A}_\epsilon$ such that $\mathbf{B}_{\mathbb{A}_\epsilon} \leq \mathbf{B} + \epsilon$, which means that $\mathbf{B} + \epsilon$ is an upper bound. Hence $\inf U \leq \mathbf{B}$. □

We are interested in *optimal algorithms* $\mathbb{A}$ such that $\mathbf{B} = \mathbf{B}_{\mathbb{A}}$. The optimality depends of course on the protocol: even if the problem class $(\mathcal{I}, \mathcal{O})$ is fixed, optimality may change from a protocol to a more restrictive one. We will be mostly concerned with optimality for protocols that only accept online algorithms.

Proposition 2.2 (**Comparing different problem classes**). *Let $\mathcal{I}' \supset \mathcal{I}$ and $\mathcal{O}' \subset \mathcal{O}$ be the sets of inputs and outputs. If the protocol for the problem class $(\mathcal{I}', \mathcal{O}')$ does not impose less constraints on the output than the protocol for the problem class $(\mathcal{I}, \mathcal{O})$ then the tight bound for said protocol for the problem class $(\mathcal{I}, \mathcal{O})$ is not higher than the tight bound for said protocol for the problem class $(\mathcal{I}', \mathcal{O}')$:*

$$\mathbf{B}(\mathcal{I}, \mathcal{O}) \leq \mathbf{B}(\mathcal{I}', \mathcal{O}') \,.$$

This Proposition provides a tool to estimate the bounds by reducing the study of a protocol to a known or simpler one. One of its applications is when the outputs are not only the corners of the inputs, but also some internal points. In such a case skipping the internal points gives a (possibly higher) estimate. Another possible application is to estimate the bound using a bound for a problem class with larger set of outputs and no extra constraint on the acceptable set of algorithms.

Proof of Proposition 2.2. The condition on constraints implies that every admissible algorithm for $(\mathcal{I}', \mathcal{O}')$ induces by restriction an admissible algorithm for $(\mathcal{I}, \mathcal{O})$. If U is an upper bound for $(\mathcal{I}', \mathcal{O}')$ there exists an algorithm $\mathbb{A}$ with $B_{\mathbb{A}}(\mathbf{\Lambda}) \leq U$, for all $\mathbf{\Lambda} \in \mathcal{I}'^{\mathbb{N}}$, therefore its restriction to $\mathbf{\Lambda} \in \mathcal{I}^{\mathbb{N}}$ also satisfies this inequality. It follows that U is an upper bound for $(\mathcal{I}, \mathcal{O})$. Hence $\inf U$ for $(\mathcal{I}, \mathcal{O})$ is not larger than $\inf U$ for $(\mathcal{I}', \mathcal{O}')$. $\square$

Notice that similarly every lower bound L for $(\mathcal{I}, \mathcal{O})$ is also a lower bound for $(\mathcal{I}', \mathcal{O}')$, hence $\sup L$ for $(\mathcal{I}', \mathcal{O}')$ is not smaller than $\sup L$ for $(\mathcal{I}, \mathcal{O})$.

From now on, we assume that all algorithms are online, except otherwise stated.

Proposition 2.3 (**The lower bound stays forever**). *Let for given algorithm $\mathbb{A}$ and demand $\mathbf{\Lambda}$:*

$$K_t = K_t(\mathbb{A}, \mathbf{\Lambda}) = \sup_{s \le t} \max_{1 \le j \le n} |E_j(s)| .$$

Then for every $\mathbf{\Lambda}$ and every t there exists $\mathbf{\Lambda}'$ such that for every w we have

$$K'_w = K_w(\mathbb{A}, \mathbf{\Lambda}') \ge K_t(\mathbb{A}, \mathbf{\Lambda}) .$$

Proof. Let $v \le t$ and $1 \le j \le n$ be such that $K_t = |E_j(v)|$. Let $\mathbf{\Lambda}'(s) = \mathbf{\Lambda}(s)$ for $s \le v$ and $\mathbf{\Lambda}'(s) = \mathbf{e}_j$ for $s > v$. Then because we are dealing with on-line algorithms for $w \le v$ we have with obvious notation $\mathbb{T}(w) = \mathbb{T}'(w)$ and $K'_w = K_w$, while for $w > v$ we have $K'_w \ge |E'_j(w)| = |E'_j(v)| = |E_j(v)| = K_t$, as for $w > v$ we were adding (input) and subtracting (output) 1 in $E'_j(w)$. $\square$

Proposition 2.4. *For any greedy algorithm with no restrictions in choosing the task (as in CAP) $E_j(t) > -1 + \frac{1}{n}$.*

Proof. As the entries Θ_j of the modified input at time t sum up to 1, at least one of these entries is not smaller than $1/n$. There is no restriction on the task to be chosen, hence picking a task with a maximal entry yields to the error not less than $-1 + 1/n$. We conclude by induction, since the first error has been chosen to be zero. $\square$

Proposition 2.5 (**Sum of errors**). *Let us assume that the chosen task has a modified input which is not smaller than any modified input with nonzero demand (this covers greedy algorithms in both the chairman assignment and the carpool problems classes). For a subset of indices $I \subset \{1, \dots, n\}$ denote $S_I(t) = 2\sum_{i \in I} E_i(t)$. Then:*

$$|S_I(t)| \le k(n-k) ,$$

where $k = \#I$ is the cardinality of I.

Proof. By construction, the errors satisfied the condition

$$S_{\{1,\dots,n\}}(t) = \sum_i E_i(t) = 0$$

and by definition $S_\emptyset(t) = 0$, which covers the cases $k = n$ and $k = 0$.

For other k's, we offer a proof by induction on time t. The conclusion is clearly true for $t = 0$, since $\mathbf{E}_i(0) = 0$ for all i. It is then enough to prove that for any k the sum is at least $-k(n-k)$ as by symmetry of the formula the reminder sum cannot exceed $(n-k)k$. Suppose that the statement holds up to $t-1$ for all k's. Let $I \subset \{1, \dots, n\}$ with $k = \#I > 0$. If the index of the chosen task at time t, $j = j(t) \notin I$ we are done, as then $S_I(t) \ge S_I(t-1)$, because we were adding the demand and we did not subtract 1 for any $i \in I$. We assume hence that $j \in I$, and let $I^- = I \setminus \{j\}$ with $\#I^- = k - 1 \ge 0$. For a moment we shall suppress the dependence on time $t-1$ and will write $E_i = E_i(t-1)$ and $S_I = S_I(t-1)$. By induction we have that:

$$S_{I^-} \geq -(k-1)(n-k+1) = -k(n-k) + (n-2k+1) \tag{2.1}$$

We denote the demand vector at time $t-1$ by λ and we define $L_I = 2\sum_{i\in I}\lambda_i$. Let $J = \{i \notin I : \lambda_i > 0\}$ with $m = \#J$ and $J^+ = J \cup \{j\}$. Clearly $0 \leq k+m \leq n$. Remark that:

$$2 = L_I + L_J = L_{I^-} + 2\lambda_j + L_J = L_{I^-} + L_{J^+} \tag{2.2}$$

By assumption the modified input for the chosen task prevails over all tasks with positive demand:

$$E_j + \lambda_j \geq E_i + \lambda_i \,,$$

and in particular for all $i \in J$. From $S_{I\cup J} = S_{I^-} + 2E_j + S_J$ we have:

$$\begin{aligned} S_{I^-} + 2(m+1)(E_j + 2\lambda_j) &\geq S_{I^-} + 2\textstyle\sum_{i\in J^+}(E_i + 2\lambda_i) \\ &\geq S_{I^-} + 2E_j + S_J + L_{J^+} \\ &= S_{I\cup J} + L_{J^+} \\ &\geq -(k+m)(n-k-m) + L_{J^+} \\ &= -k(n-k) - m(n-2k-m) \\ &\quad + L_{J^+} \end{aligned} \tag{2.3}$$

Adding $\frac{m}{m+1}$ times inequality (2.1) to $\frac{1}{m+1}$ times inequality (2.3) we get:

$$\begin{aligned} S_I + 2\lambda_j &= S_{I^-} + 2E_j + 2\lambda_j \\ &\geq -k(n-k) + m + \tfrac{L_{J^+}}{m+1} \\ &= -k(n-k) + m + \tfrac{2-L_{I^-}}{m+1} \,, \end{aligned}$$

where we used (2.2). We return to the time depending notation. By equations (1.4) we have $S_I(t) = S(t-1) + L_I - 2$ and by definition $L_I - 2\lambda_j = L_{I^-} \geq 0$. Then:

$$\begin{aligned} S_I(t) &\geq -k(n-k) + m + \frac{2-L_{I^-}}{m+1} - 2\lambda_j + L_I - 2 \\ &= -k(n-k) + m - 2 + \frac{2}{m+1} + L_{I^-}(1 - \frac{1}{m+1}) \\ &\geq -k(n-k) \,, \end{aligned}$$

as $m + 2/(m+1) \geq 2$ for all natural numbers m. When $m = 0$ then $L_I = 2$ and $S_I(t) = S_I(t-1)$. □

3. The Chairman Assignment Problem (CAP)

This problem, first presented in [9] in the constraint-less protocol version (although it is so basic that we may have missed former formulations), was described in Section 1 as pertaining to the problem class for which $\mathcal{I} = \Delta$, $\mathcal{O} = \{\mathbf{e}_j\}$. In the original protocol, one seeks the absolute best solutions and there is no restriction on the demands and tasks: see for instance [9], [11] , [8], [12]. We are here concerned with the protocol characterized by restricting to online algorithms, for which some remarks were made by Tijdeman in [12]. We use Proposition 2.4 to restrict our consideration to positive values of the errors. Denote the harmonic series by:

$$H(n) = \sum_{j=1}^{n} \frac{1}{j} \,.$$

Recall that for $n > 1$, $H(n) - 1 < \ln(n) < H(n-1)$.

Theorem 3.1 (**Tight bound for CAP**). *Let $\mathcal{I} = \Delta$, $\mathcal{O} = \{\mathbf{e}_j\}$ and there are no other conditions on the protocol (Chairman Assignment Problem). Then for on-line algorithms:*

$$\mathbf{B} = H(n) - 1 \,.$$

Proof. **Lower bound:** $\sup L \geq H(n) - 1$. This was shown in [12]. It is enough to prove that $\sum_{j=2}^{n} \frac{1}{j}$ is a lower bound. We shall prove that for any algorithm this number is achieved it at most n steps on a specific family of demands. We consider the demands which at time $t = 1, \dots, n$ have either entries 0 or $\frac{1}{n-t+1}$, because they must add to 1 there are $n - t + 1$ positive, equal entries and $t - 1$ zeros. In particular $\mathbf{\Lambda}(1) = (\frac{1}{n}, \dots, \frac{1}{n})$ and $\mathbf{\Lambda}(n) = \mathbf{e}_j$ for some $1 \leq j \leq n$.

Let fix an algorithm and let $N(t)$ be an increasing family of the sets of t indices which contain the indices of the tasks chosen up to time t; if for some $s \leq t$ we have $\mathbb{T}(s) = \mathbb{T}_j$ then $j \in N(t)$. At each time $1 < t \leq n$ there is a demand from our family such that the set of indices of zeros of $\mathbf{\Lambda}(t)$ is equal to $N(t)$. Therefore for $j \notin N(t)$ the algorithm did not choose $\mathbb{T}_j$ up to time t, there were no substraction of 1, but on the other hand there were always an addition of a positive entry and hence $E_j(t) = \sum_{s=1}^{t} \frac{1}{n-s+1}$. At time $n - 1$ we have $N(n-1) \leq n - 1$ and there is at least one entry with $E_j(n-1) = \sum_{s=1}^{n-1} \frac{1}{n-s+1} = \sum_{j=2}^{n} \frac{1}{j}$. That means that for every algorithm $\mathbb{A}$:

$$\mathbf{B}_{\mathbb{A}} = \sup_{\mathbf{\Lambda}} B_{\mathbb{A}}(\mathbf{\Lambda}) = \sup_{\mathbf{\Lambda}} \sup_{t} \sup_{j} E_j(t) \geq \sum_{j=2}^{n} \frac{1}{j} \,,$$

or that $H(n) - 1$ is a lower bound.

Upper bound $\inf U \leq H(n) - 1$. We prove this upper bound for the greedy algorithm. Let us fix a time moment t, and a task $\mathbb{T}$. At this moment we rename the tasks by $\mathbb{T}_1 = \mathbb{T}$ and the other ones in the order which they were last chosen. So $\mathbb{T}_2$ is the most recently chosen task, $\mathbb{T}_3$ is the most recently chosen task besides $\mathbb{T}_2$, etc. It is possible that some tasks were never chosen up to time t. For $j \geq 2$ we call t_j the moment when $\mathbb{T}_j$ was last chosen, we have hence $t = t_2$ and t_j is decreasing. With this convention we have $\mathbf{e}(t_j) = \mathbf{e}_j$.

For $2 \leq k \leq j$, let $z_j^k = E_j(t_k)$ be the error of task $\mathbb{T}_j$ just after the time when task $\mathbb{T}_k$ was chosen for the last time. Notice that when $k < j$ we have $t_k > t_j$, and as $\mathbb{T}_j$ was not chosen after t_j any algorithm only adds to its error non negative demand entries and never subtracts 1, so this error cannot decrease. So for any $j \geq k' \geq k \geq 2$ we have:

$$z_j^j \leq z_j^{k'} \leq z_j^k \leq z_j^2 \,. \tag{3.1}$$

Consider the modified input vector at time t_k, $\mathbf{\Theta}(t_k) = \mathbf{E}(t_k - 1) + \mathbf{\Lambda}(t_k)$ whose entries sum up to 1. As the algorithm is greedy we have $\Theta_j(t_k) \leq \Theta_k(t_k)$ for all j, in particular for $j < k$. For $j > k$ the tasks were not chosen and we have $\Theta_j(t_k) = z_j^k$. Hence for $k \geq 2$:

$$1 = \sum_{j=1}^{n} \Theta_j(t_k) \leq k\Theta_k(t_k) + \sum_{j=k+1}^{n} z_j^k \,.$$

The error of $\mathbb{T}_k$ after it was chosen is hence:

$$E_k(t_k) = z_k^k = \Theta_k(t_k) - 1 \geq \frac{1}{k}(1 - \sum_{j=k+1}^{n} z_j^k) - 1\,.$$

Multiplying this inequality by $\frac{1}{k-1}$ we obtain $\frac{1}{k(k-1)}\sum_{j=k+1}^{n} z_j^k + \frac{1}{k-1} z_k^k \geq -\frac{1}{k}$. If $\mathbb{T}_k$ has never been assigned, then $\mathbb{T}_{k+1}, \ldots, \mathbb{T}_n$ have also never been assigned and the preceding inequality still holds, since all values on the left-hand side are nonnegative. By inequality (3.1), we get:

$$\frac{1}{k(k-1)} \sum_{j=k+1}^{n} z_j^2 + \frac{1}{k-1} z_k^2 \geq -\frac{1}{k}\,,$$

for $2 \leq k \leq n$. Summing over k we obtain:

$$\begin{aligned}
-H(n) + 1 &\leq \sum_{k=2}^{n} \sum_{j=k+1}^{n} \frac{z_j^2}{k(k-1)} + \sum_{k=2}^{n} \frac{1}{k-1} z_k^2 \\
&= \sum_{j=3}^{n} \sum_{k=2}^{j-1} \frac{z_j^2}{k(k-1)} + \sum_{j=2}^{n} \frac{1}{j-1} z_j^2 \\
&= \sum_{j=2}^{n} z_j^2 \left(\left(\sum_{k=2}^{j-1} \frac{1}{k-1} - \frac{1}{k} \right) + \frac{1}{j-1} \right) \\
&= \sum_{j=2}^{n} z_j^2
\end{aligned}$$

Since at each time the errors sum up to zero, in particular for time $t_2 = t$ we have $\sum_{j=1}^{n} z_j^2 = 0$. It follows that $E_1(t) = z_1^2 \leq H(n) - 1$. As t and $\mathbb{T}_1$ were arbitrary, the estimate holds for every task at any time. $\square$

Remark. *The proof of Theorem 3.1 actually shows a little more: the errors satisfy* $E_j \in [\frac{1}{n} - 1, H_n - 1]$ *for the greedy on-line algorithm, and this is a tight bound. The upper endpoint is achieved by the example in the proof of Theorem 3.1, and the lower endpoint is achieved with* $\lambda(1) = (\frac{1}{n}, \ldots, \frac{1}{n})$.

Remark. *For off-line algorithms,* $\mathbf{B} = 1 - \frac{1}{2(n-1)}$ [8, 11, 12] *(tight bound). In other words, whereas for on-line algorithms,* $\mathbf{B}$ *grows as* $\ln(n)$*, there exists an off-line algorithm for which* $\mathbf{B}$ *is less than* 1 *for all n (a condition which is called P-*fair *in* [5]*).*

4. The Generalized Carpool Problem (GCP)

This problem refers to the same protocol as the Chairman Assignment Problem in Section 3, except that the chosen task $\mathbb{T}_{j_t}$ must satisfy the additional requirement that $\lambda_{j(t)} > 0$, *i.e.,* the task chosen must be among those tasks whose demand is nonzero. From its name, the reader will guess that GCP is also related to the Carpool Problem (CP) that is studied in Section 5. We came in fact to formulate GCP by generalizing CP, seeing of course right away that the generalization of CP we had in hand was related to CAP. We then realized that GCP makes a lot of sense in actual chairman assignment situations, and more precisely, each time one needs the chairman to be present to do the job.

Theorem 4.1 (Tight bound in GCP). *Let $\mathcal{I} = \Delta$, $\mathcal{O} = \{\mathbf{e}_j\}$ and if a coordinate of an input at some time is zero, so is the corresponding coordinate of the output at the same time (Generalized Carpool Problem). Then*

$$\mathbf{B} = \frac{n-1}{2}\,.$$

Proof.
Lower bound: $\sup L \geq \frac{n-1}{2}$.
It is enough to prove a slightly stronger statement: for every on-line algorithm $\mathbb{A}$ solving the GCP there is a demand sequence $\mathbf{\Lambda}$ such that $\limsup_t B_{\mathbb{A}}(\mathbf{\Lambda}) \geq (n-1)/2$.

For any time t let $l = l(t)$ and $l' = l'(t)$ be such two different indices that the gap $g = g(t) = E_l(t) - E_{l'}(t) \geq 0$ is minimal. Then $\max_{1\leq i,j\leq n} E_i(t) - E_j(t) \geq (n-1)g$ and therefore:

$$\max_i |E_i(t)| \geq (n-1)g(t)/2\,. \tag{4.1}$$

Having this we define inductively a demand sequence by $\lambda_{l'}(t) = (1+g(t))/2$, $\lambda_l(t) = (1-g(t))/2$, and all other $\lambda_i(t) = 0$. By the carpool protocol only task $\mathbb{T}_l$ or $\mathbb{T}_{l'}$ can be chosen. The modified inputs of both of them are equal to $(E_l(t)+E_{l'}(t)+1)/2 = E_l(t) + (-g(t)+1)/2$, as by definition $E_{l'} = E_l - g$. Hence for $i \neq l, l'$ the entries of the error vector at time $t+1$ are $E_i(t+1) = E_i(t)$ and the two remaining entries are $E_l(t)+(-g(t)+1)/2$ and $E_l(t)+(-g(t)-1)/2$ (for the chosen one we subtracted 1). Define the second moment of the error vector by $D(t) = \sum_{1\leq i\leq n} E_i(t)^2$. We notice that if $D(t) \geq A^2$ then at least one $|E_i(t)| \geq A/\sqrt{n}$. Then:

$$\begin{aligned} D(t+1) - D(t) &= (E_l - \tfrac{g-1}{2})^2 + (E_l - \tfrac{g+1}{2})^2 \\ &\quad -(E_l^2 + (E_l - g)^2) \\ &= \tfrac{1-g^2}{2} \end{aligned} \tag{4.2}$$

We have three possibilities:

(1) $g(t) \geq 1$ for infinitely many t, then, by (4.1), $\|E_i(t)\|_\infty \geq (n-1)/2$ infinitely often and hence this holds for
$$\limsup_t \|\mathbf{E}(t)\|_\infty$$
(2) $g(t) < 1$ for almost every t, but $g(t) \geq \sqrt{1-2/t}$ infinitely many times. Then again by (4.1)
$$\limsup_t \|\mathbf{E}(t)\|_\infty \geq (n-1)/2$$
(3) $g(t) < \sqrt{1-2/t}$ for almost every t, in particular there exists a t_0 such that it holds for all $s \geq t_0$. Then by (4.2) for any $t_1 \geq t_0$ there is a $t > t_1$ such that $D(t) \geq D(t_1) + \sum_{t_1\leq s<t} \frac{1}{s} \geq n(n-1)^2/2$. Thus $\|\mathbf{E}(t)\|_\infty \geq (n-1)/2$ for infinitely many t. Notice that in fact in this case $\limsup_t \|\mathbf{E}(t)\|_\infty = \infty$.

Upper bound: $\inf U \leq \frac{n-1}{2}$
We estimate the upper bound for the greedy algorithm, so that we can use Proposition 2.5, where we pick the case $k = 1$. □

In fact, this bound is only approached asymptotically. In particular,

Theorem 4.2. *Consider the greedy algorithm with $\mathcal{O} = \{\mathbf{e}_j\}$ and $n > 2$. For $1 \leq k \leq n-1$, the magnitude of the sum of any collection of k errors E_j's is strictly less than $\frac{k(n-k)}{2}$. This implies that $\|\mathbf{E}(t)\|_\infty < \frac{n-1}{2}$.*

Proof. From Proposition 2.5 we know that $\sum_{j=1}^{k} E_j \geq -\frac{k(n-k)}{2}$ for $1 \leq k < n$. To reach a contradiction, consider the first time t such that:

$$\sum_{j=1}^{k} E_j = -\frac{k(n-k)}{2}\,, \tag{4.3}$$

for some k, $1 \leq k < n$. Suppose also that k is the largest such integer. The chosen task $\mathbb{T}_d$ must have been among these k tasks and at least one of the tasks with nonzero demands was not among the k tasks, say task $\mathbb{T}_p$. We consider the group of tasks without the chosen task and use again Proposition 2.5:

$$E_1 + \cdots + E_k - E_d \geq -\frac{(k-1)(n-k+1)}{2}\,,$$

and subtracting equation (4.3) we get $E_d \leq -\frac{n-2k+1}{2}$.

Considering the k tasks plus $\mathbb{T}_p$, we have $E_1 + \cdots + E_k + E_p \geq -\frac{(k+1)(n-k-1)}{2}$. Subtracting equation (4.3) we get $E_p \geq -\frac{n-2k-1}{2}$. Therefore the modified input for task $\mathbb{T}_d$ is less than or equal to the modified input for task $\mathbb{T}_p$, *i.e.,* $E_d + 1 \leq E_P$. If $k > 1$, consider the $k-1$ tasks without the chosen task $\mathbb{T}_d$:

$$E_1 + \cdots + E_k - E_d > \frac{(k-1)(n-k+1)}{2}\,,$$

which holds at time $t-1$ and continues to holds for time t. Therefore $E_d < -\frac{n-2k+1}{2}$. If $k < n-1$, then by maximality of k, $(E_1 + \cdots + E_k + E_p) > -\frac{(k+1)(n-k-1)}{2}$ which implies $E_p > -\frac{n-2k-1}{2}$. In either case ($k > 1$ or $k < n-1$) we have $E_d + 1 < E_p$, which means that task $\mathbb{T}_d$ should not have been chosen under the greedy algorithm. The only possibility to avoid a contradiction is if $k = 1 = n-1$, *i.e.,* $n = 2$. □

5. The Carpool Problem (CP)

This problem, first proposed in [6], is the same as the generalized carpool problem with the additional constraint that the demand vector $\mathbf{\Lambda}(t)$ is such that for some integer b (which may depend on t), b of its elements are equal to $\frac{1}{b}$ and the rest are zero.

Theorem 5.1 (Lower bound for B in CP).

$$\mathbf{B} > \frac{3(n-1)}{8} - \frac{1}{4}\,.$$

Proof. We will only use demand vectors with 2 or 3 nonzero entries and show that this bound is true for any on-line algorithm. Note that the error will be a multiple of $\frac{1}{6}$. Again let $(E_1, \ldots, E_n)$ denote the error $\mathbf{E}(t)$. Recall that $\sum E_j = 0$. Consider the error vector $\mathbf{E}(t)$ maximizing the second moment $\sum E_j^2$ among all error vectors reachable from $\mathbf{E}(0) = (0, 0, ..., 0)$ after a finite time using only demand vectors with either 2 or 3 nonzero entries. Such an error vector exists since the second moment can only attain a finite number of values (as the second moment is a multiple of $\frac{1}{36}$), and if the second moment were to become unbounded, then the theorem is trivially true. We assume that E_j are arranged in ascending order. For this maximal error vector, the gap g between two successive E_j is at least $\frac{3}{6}$. Otherwise we can create a demand vector where these two tasks have demand $\frac{1}{2}$ and increase the second moment: if $0 \leq g < 1/2$ then both $(E_j + 1/2)^2 + (E_j + g - 1/2)^2 E_j^2 + (E_j + g)^2$ and $(E_j - 1/2)^2 + (E_j + g + 1/2)^2 > E_j^2 + (E_j + g)^2$. Here the first inequality represents the result of applying the greedy algorithm, while the second represents the other choice. So each gap is at least 1/2, which gives the lower bound of $\frac{n-1}{4}$ that was given in [6].

Next we show that two successive gaps cannot both be $\frac{3}{6}$. Suppose that

$$(E_j, E_{j+1}, E_{j+2}) = (E_j, E_j + \frac{3}{6}, E_j + \frac{6}{6})\,.$$

Using a demand vector giving all three of these tasks a demand of $\frac{1}{3}$, the error at the next time is then $(E_j + \frac{2}{6}, E_j + \frac{2}{6}, E_j + \frac{5}{6})$ (if the greedy algorithm is followed). Next a demand vector for the first two tasks each with a demand of $\frac{1}{2}$ results in the errors $(E_j - \frac{1}{6}, E_j + \frac{5}{6}, E_j + \frac{5}{6})$ (no matter which algorithm is followed, since the two candidates have equal errors). The latter two tasks next have demand $\frac{1}{2}$, resulting in the error $(E_j - \frac{1}{6}, E_j + \frac{2}{6}, E_j + \frac{8}{6})$, which has larger second moment. If the greedy algorithm had not been followed, the first step would have led to errors $(E_j - \frac{4}{6}, E_j + \frac{5}{6}, E_j + \frac{8}{6})$ or $(E_j - \frac{1}{6}, E_j + \frac{2}{6}, E_j + \frac{8}{6})$, each of which already has higher second moment.

Other patterns of consecutive small gaps can be treated similarly. Here we list a pattern of consecutive small gaps, the related initial error pattern, and a pattern which results after several iterations of presenting a carefully chosen demand vector and following the greedy algorithm. In each case this latter pattern has a larger second moment than the initial pattern, and in each case one can check that any other online algorithm would also lead to an increase in second moment, (see Appendix B for more details).

Gaps	Initial errors
$(\frac{0}{6})$	$(E, E + \frac{0}{6})$
$(\frac{1}{6})$	$(E, E + \frac{1}{6})$
$(\frac{2}{6})$	$(E, E + \frac{2}{6})$
$(\frac{3}{6}, \frac{3}{6})$	$(E, E + \frac{3}{6}, E + \frac{6}{6})$
$(\frac{3}{6}, \frac{4}{6})$	$(E, E + \frac{3}{6}, E + \frac{7}{6})$
$(\frac{4}{6}, \frac{3}{6})$	$(E, E + \frac{4}{6}, E + \frac{7}{6})$
$(\frac{3}{6}, \frac{5}{6})$	$(E, E + \frac{3}{6}, E + \frac{8}{6})$
$(\frac{5}{6}, \frac{3}{6})$	$(E, E + \frac{5}{6}, E + \frac{8}{6})$
$(\frac{3}{6}, \frac{6}{6}, \frac{3}{6})$	$(E, E + \frac{3}{6}, E + \frac{9}{6}, E + \frac{12}{6})$
$(\frac{3}{6}, \frac{6}{6}, \frac{4}{6})$	$(E, E + \frac{3}{6}, E + \frac{9}{6}, E + \frac{13}{6})$
$(\frac{4}{6}, \frac{6}{6}, \frac{3}{6})$	$(E, E + \frac{4}{6}, E + \frac{10}{6}, E + \frac{13}{6})$
$(\frac{4}{6}, \frac{4}{6}, \frac{4}{6})$	$(E, E + \frac{4}{6}, E + \frac{8}{6}, E + \frac{12}{6})$
$(\frac{4}{6}, \frac{4}{6}, \frac{5}{6})$	$(E, E + \frac{4}{6}, E + \frac{8}{6}, E + \frac{13}{6})$
$(\frac{5}{6}, \frac{4}{6}, \frac{4}{6})$	$(E, E + \frac{5}{6}, E + \frac{9}{6}, E + \frac{13}{6})\,.$

$$\begin{array}{c}
\text{final errors} \\
(E-\frac{3}{6},E+\frac{3}{6}) \\
(E-\frac{2}{6},E+\frac{3}{6}) \\
(E-\frac{1}{6},E+\frac{3}{6}) \\
(E-\frac{1}{6},E+\frac{2}{6},E+\frac{8}{6}) \\
(E-\frac{1}{6},E+\frac{3}{6},E+\frac{8}{6}) \\
(E-\frac{1}{6},E+\frac{4}{6},E+\frac{8}{6}) \\
(E-\frac{1}{6},E+\frac{4}{6},E+\frac{8}{6}) \\
(E+\frac{0}{6},E+\frac{4}{6},E+\frac{9}{6}) \\
(E-\frac{1}{6},E+\frac{4}{6},E+\frac{8}{6},E+\frac{13}{6}) \\
(E-\frac{1}{6},E+\frac{4}{6},E+\frac{9}{6},E+\frac{13}{6}) \\
(E+\frac{0}{6},E+\frac{4}{6},E+\frac{9}{6},E+\frac{14}{6}) \\
(E-\frac{1}{6},E+\frac{4}{6},E+\frac{8}{6},E+\frac{13}{6}) \\
(E-\frac{1}{6},E+\frac{4}{6},E+\frac{9}{6},E+\frac{13}{6}) \\
(E+\frac{0}{6},E+\frac{4}{6},E+\frac{9}{6},E+\frac{14}{6}).
\end{array}$$

For example, the gaps $(\frac{3}{6},\frac{6}{6},\frac{4}{6})$ and errors $(E,E+\frac{3}{6},E+\frac{9}{6},E+\frac{13}{6})$ (which we will abbreviate as (0,3,9,13)) go through the sequence $(0,3,9,13)\to(2,5,5,13)\to(2,2,8,13)\to(-1,5,8,13)\to(-1,7,10,9)\to(-1,7,7,12)\to(-1,4,10,12)\to(-1,4,13,9)=(-1,4,9,13)$ under the greedy algorithm. One can check that at each stage, a departure from the choices of the greedy algorithm would lead even faster to an increase in second moment.

The error vector maximizing the second moment contains none of these sequences of consecutive small gaps. Starting at an arbitrary location, one can check that either its first gap is at least $\frac{5}{6}$, or the sum of the first two gaps is at least $\frac{9}{6}$, or the sum of the first three gaps is at least $\frac{14}{6}$. Proceeding left-to-right, we conclude that the average value of all but the last one or two gaps is at least $3/4=(\frac{9}{6})/2$. Thus $E_n-E_1\geq\min\{(n-3)(3/4)+\frac{4}{6}+\frac{4}{6},(n-2)(3/4)+\frac{3}{6}\}=\frac{3(n-1)-1}{4}$. Similarly $E_n-E_j\geq\frac{3(n-j)-1}{4}$, so that:

$$\begin{aligned}
nE_n &= (n-1)E_n-\sum_{j=1}^{n-1}E_j=\sum_{j=1}^{n-1}(E_n-E_j) \\
&\geq \sum_{j=1}^{n-1}\frac{3(n-j)-1}{4}=\frac{3n(n-1)-2(n-1)}{8} \\
\mathbf{B}\geq E_n &> \frac{3(n-1)-2}{8}.
\end{aligned}$$

The most extreme example is apparently when consecutive gaps alternate between $\frac{4}{6}$ and $\frac{5}{6}$. □

Theorem 5.2 (Upper bound for B in CP). *If $n>2$, then:*

$$\mathbf{B}\leq\frac{n-1}{2}-\frac{1}{\operatorname{lcm}(2,3,\ldots,n)}.$$

where $\operatorname{lcm}(a_1,\ldots a_m)$ is the least common multiple of the set of numbers a_i.

Proof. It follows from Theorem 4.2 that at each time $\|\mathbf{E}(t)\|_\infty<\frac{n-1}{2}$ for the greedy algorithm. Since each element of the demand vector (and thus also $\mathbf{E}(t)$) is a multiple of $\frac{1}{\operatorname{lcm}(2,3,\ldots,n)}$ the result follows. □

Remark. *Theorem 5.2 was announced in* [6]. *For an alternate proof, see* [4].

6. The Chairman Assignment Problem with Multiple Tasks (MCAP)

Consider now the case where m tasks are chosen, but within each time step, the same task cannot be chosen more than d times, for instance if $d = 1$, all the chosen tasks at each time are distinct. Using the notation as in Section 1 the demand vector $\mathbf{\Lambda}(t)$ satisfies $0 \leq \lambda_j(t) \leq d$ and $\sum_j \lambda_j(t) = m$ where m and d are integers with $1 \leq d \leq m$. At each time m tasks are chosen such that the same task cannot be chosen more than d times. Extending the notation from the introduction the output $\mathbf{c}(t)$ is now a point $\sum_j \alpha_j \mathbf{c}_j$ where α_j are integers, $0 \leq \alpha_j \leq d$, and $\sum_j \alpha_j = m$. The greedy algorithm for this case is defined as the algorithm where at each time step, the $\lfloor \frac{m}{d} \rfloor$ tasks with the largest modified inputs are each chosen d times and the task with the next largest modified input is chosen m mod d times.

Remark. *Using the convention from the Section 1 we have the demands coming from the polytope which is the convex hull of the set of the following points (outputs):*

$$\mathcal{C} = \left\{ \mathbf{c} \in \{0, \ldots, d\}^n : \sum_{j=1}^{n} \mathbf{c}_j = m \right\} .$$

Notice that when $d > 1$, not all the outputs are the corners (extremities) of the polytope. For $d = 1$ the corners are subsets of the corners of the standard cube $\{0,1\}^n$ and the polytopes are called hyper-simplices.

If we allow the non-integer coefficients α_j then we can consider the following MCAP formulation. The inputs are from the polytope:

$$\mathcal{I} = \left\{ p : \sum_j p_j = \sigma = \frac{m}{d}, p_j \in [0,1] \right\} ,$$

and the set $\mathcal{O}$ of outputs form the set $\mathcal{O}_M CAP$ of all the rational points of the polytope $\mathcal{I}$ with the denominator d. *A set of corners $\mathcal{C}$ of the polytope is a subset of its points with at most one of coordinate not in* $\{0,1\}$. *The upper bound for the context with outputs $\mathcal{C}$ is also an upper bound for the context with outputs in the extreme points of the convex hull of $\mathcal{C}$.*

This can be generalized to any real $\sigma \in (0, n)$.

When $d \mid m$, this reduces to the case of $\frac{m}{d}$ tasks with distinct chosen tasks since the demand vector can be divided by d and the greedy algorithm for $\frac{m}{d}$ tasks with distinct chosen tasks is applied d times. In particular, when $d = m$, this reduces to the single task case, and thus:

$$\text{if } d = m, \text{ then } \mathbf{B} \leq H(n) - 1 .$$

When $d = 1$, we can use the substitutions $\mathbf{E}(t) \to -\mathbf{E}(t)$, $\mathbf{\Theta}(t) \to (1, \ldots, 1) - \mathbf{\Theta}(t)$, $\mathbf{\Lambda}(t) \to (1, \ldots, 1) - \mathbf{\Lambda}(t)$, $\mathbf{e}_j(t) \to (1, \ldots, 1) - \mathbf{e}_j(t)$ to show that if each component of the error vector is shown to lie in the interval $[a, b]$ for the case of m tasks, then each component of the error vector lie in the interval $[-b, -a]$ for the case of $n - m$ tasks. Thus the bound $\mathbf{B}$ is the same for m tasks as it is for $n - m$ tasks. In particular, by the results in Section 3:

$$\text{If } d = 1 \text{ and } m = n - 1, \quad \text{then} \quad \mathbf{B} = H(n) - 1 .$$

Theorem 6.1 (Lower bound for B in MCAP).

$$\mathbf{B} \geq H\left(\left\lceil \frac{n}{m} \right\rceil\right) - 1 \,.$$

Proof. The proof is essentially the same as that of the lower bound in Theorem 3.1. Choose the first demand vector as $(\frac{m}{n}, \frac{m}{n}, \ldots, \frac{m}{n})$. Without loss of generality, the first m tasks are chosen. The next demand vector is chosen as $(0, \ldots, 0, \frac{m}{n-m}, \ldots, \frac{m}{n-m})$. Without loss of generality, tasks $\mathbb{T}_{m+1}, \ldots, \mathbb{T}_{2m}$ are chosen etc. This process is repeated $\lceil \frac{n}{m} \rceil - 1$ times, at which point, task $\mathbb{T}_n$ will have error $\frac{m}{n} + \frac{m}{n-m} + \cdots + \frac{m}{n-m(\lceil \frac{n}{m} \rceil - 2)} = \sum_{j=0}^{\lceil \frac{n}{m} \rceil - 2} \frac{1}{\frac{n}{m} - j}$. □

Corollary 6.2. *If $d = 1$, then $\mathbf{B} \geq H(\lceil \frac{n}{m} \rceil) - 1$ for $m \leq \frac{n}{2}$ and $\mathbf{B} \geq H(\lceil \frac{n}{n-m} \rceil) - 1$ for $m \geq \frac{n}{2}$.*

Proof. Follows from Theorem 6.1 and the discussion above. □

Theorem 6.3 (Upper bound in MCAP, case $d = 1$). *Let:*

$$\alpha = \frac{m}{n} - \frac{m}{2} - \frac{1}{2} < 0 \quad \textit{and} \quad \beta = \frac{n}{2} + \frac{m}{n} - \frac{m}{2} - \frac{1}{2} > 0 \,.$$

If $d = 1$, then the error vector $\mathbf{E}(t) = (E_1, \ldots E_n)$ satisfies $E_j \in [\alpha, \beta]$ and:

$$\mathbf{B} \leq \max\left(-\alpha, \beta\right) \,.$$

Proof. It is easy to show that $\alpha = -1 + (1 - (m-1)\beta)/(n - m + 1)$, $\beta = \alpha + \frac{n}{2}$ and $\beta(m+1) = m - (n - m - 1)\alpha$.

We prove this by induction on time t using the greedy algorithm. For $t = 0$ the errors are all zero. By the induction hypothesis the errors at time $t - 1$ are in $[\alpha, \beta]$. Therefore at time t, the modified inputs $\Theta_j \in [\alpha, \beta + 1]$.

Let J be the set of indices of the m tasks with the largest modified input. For $j \in J$, $E_j = \Theta_j - 1$, and $E_j \leq \beta$. Let $i \in J$, then:

$$\begin{array}{rcl} m & = & \sum_j \Theta_j = \sum_{j \notin J} \Theta_j + \Theta_i + \sum_{j \neq i, j \in J} \Theta_j \\ & \leq & (n - m + 1)\Theta_i + (m-1)(\beta + 1) \,. \end{array}$$

Therefore:

$$E_i = \Theta_i - 1 \geq \frac{m - (m-1)(\beta + 1)}{n - m + 1} - 1 = -1 + \frac{1 - (m-1)\alpha}{n - m + 1} = \alpha \,.$$

On the other hand for $j \notin J$, $E_j = \Theta_j$, hence $E_j \geq \alpha$. It remains to show the upper bound. Let $k \notin J$. Then $\Theta_k \leq \Theta_j$ for all $j \in J$, and:

$$\begin{array}{rcl} m & = & \sum_j \Theta_j = \sum_{j \notin J, j \neq k} \Theta_j + \Theta_k + \sum_{j \in J} \Theta_j \\ & \geq & (n - m - 1)\alpha + (m + 1)\Theta_k \,. \end{array}$$

This implies that:

$$E_k = \Theta_k \leq \frac{m - (n - m - 1)\alpha}{m + 1} = \beta \,.$$

Thus $E_j \in [\alpha, \beta]$ and the proof is complete. □

Corollary 6.4. *If $d \mid m$, then*

$$\mathbf{B} \leq \max\left(\frac{m}{2d} + \frac{1}{2} - \frac{m}{dn}, \frac{n}{2} + \frac{m}{dn} - \frac{m}{2d} - \frac{1}{2}\right)$$

In particular, the error vector $\mathbf{E}(t) = (E_1, \dots E_n)$ *satisfies* $\frac{m}{dn} - \frac{m}{2d} - \frac{1}{2} \leq E_j \leq \frac{n}{2} + \frac{m}{dn} - \frac{m}{2d} - \frac{1}{2}$.

Proof. Follows from Theorem 6.3 and the discussion above. □

For the case when d does not divide m, geometric ideas will be useful in providing an upper bound for $\mathbf{B}$. We will give such a result in Section 7.

The bounds for these problem classes are summarized in Table 1.

	lower bound of $\mathbf{B}$	upper bound of $\mathbf{B}$
CAP	$H(n) - 1$	$H(n) - 1$
GCP	$\frac{n-1}{2}$	$\frac{n-1}{2}$
CP	$\frac{3(n-1)}{8} - \frac{1}{4}$	$\frac{n-1}{2} - \frac{1}{\operatorname{lcm}(2,\dots,n)}$ (for $n > 2$)
MCAP	$H\left(\lceil \frac{n}{m} \rceil\right) - 1$	$\max\left(\frac{m+1}{2} - \frac{m}{n}, \frac{n-m-1}{2} + \frac{m}{n}\right)$ (for $d = 1$)

TABLE 1. Table of upper and lower bounds of maximal error vector norm $\mathbf{B}$.

7. Geometric Version of the Problem

The general setting for a geometric version of this problem is as follows:

Given a polytope P in $\mathbb{R}^q$ with vertices $v_1, \dots, v_n$, at each time t, a demand $\mathbf{\Lambda}(t) \in P$ is given and a vertex $v_{j(t)}$ is chosen. The error at time t is given by $\mathbf{E}(t) = \sum_{s=1}^{t} \left(\mathbf{\Lambda}(s) - v_{j(s)}\right)$. The goal is to pick vertices such that $\|\mathbf{E}(t)\|$ is minimized. The Euclidean norm $\|\cdot\|_2$ is usually chosen for this problem. As before on-line means that $v_{j(t)}$ depends only on $\mathbf{\Lambda}(t)$, $\mathbf{\Lambda}(s)$ and $v_{j(s)}$ for $s < t$. The greedy algorithm for a metric ρ on $\mathbb{R}^q$ is given by: at each time t, pick the (admissible) vertex which is closest under ρ to the modified input $\Theta(t) = \mathbf{E}(t-1) + \mathbf{\Lambda}(t)$.

Remark. *It can be shown that the greedy algorithm for the geometric version under the Euclidean metric coincide with the greedy algorithm in Section 3 when the polytope is the standard simplex* Δ. *Note also that if* $\mathbf{\Lambda}(t)$ *are allowed to be outside of* P*, then one can choose a series of* $\mathbf{\Lambda}$*'s such that* $\|\mathbf{E}(t)\|$ *is unbounded* [2].

The next result shows that a upper bound for this problem can be found from the upper bound for the CAP.

Theorem 7.1. *There exists an on-line algorithm such that:*

$$\|\mathbf{E}(t)\| \leq \sup_{\{b_j\} \in S} \left\| \sum b_j v_j \right\| \leq (n-1) \sup_{p \in P} \|p - p_*\| ,$$

where $S = \left\{ \{b_j\} : \forall j \frac{1}{n} - 1 \leq b_j \leq H(n) - 1, \sum_j b_j = 0 \right\}$ *and* $p_* = \frac{1}{n} \sum v_j$.

Proof. The proof is similar to that of Theorem 3 in [2]. Each demand vector $\mathbf{\Lambda}(t)$ can be decomposed as a convex combination of vertices, *i.e.,* $\mathbf{\Lambda}(t) = \sum_j \alpha_j v_j$. It is then clear that the coefficients α_j form a demand vector for the CAP and that the task $\mathbb{T}_{j_t}$ chosen corresponds to picking the vertex v_{j_t}. The error vector $\mathbf{E}(t)$ in this problem is formed by taking the coefficients E_j of the error vector in the CAP and forming $\sum_j E_j v_j$. By the proof of Theorem 3.1, there exists an online algorithm such that $\frac{1}{n} - 1 \leq E_j \leq \sum_{l=2}^{n} \frac{1}{l}$ and thus the first inequality follows. The second inequality is obtained by changing b_j to $\tilde{b}_j = b_j + 1 - \frac{1}{n}$. □

In Theorem 7.1 the bound depends on the number of vertices in P. Next we present a bound which depends only on the dimension q of the polytope and is thus superior to the bound in Theorem 7.1 when the number of vertices is much larger than q.

Theorem 7.2. *There is an on-line algorithm such that:*

$$\mathbf{E}(t) \in P',$$

where $P' = \{x : \frac{1}{q}x + p_0 \in P\}$, q is the dimension of P and p_0 is any point in P.

Proof. By induction on t. Since $0 \in P'$, it's true for $t = 0$. Since $\mathbf{E}(t-1) \in P'$ it can be written as $\sum \alpha_j(v_j - p_0)$ where $\alpha_j \geq 0$, $\sum \alpha_j = q$. Since $\mathbf{\Lambda}(t) \in P$, $\Theta(t) = \mathbf{E}(t-1) + \mathbf{\Lambda}(t)$ can written as $\sum \beta_j(v_j - p_0) + p_0$ where $\beta_j \geq 0$, $\sum \beta_j = q+1$. By Carathéodory's theorem β_j can be chosen such that the number of nonzero β_j's is at most $q+1$. This means that there exists a vertex v_j such that $\beta_j \geq 1$. Pick this vertex and it's clear that $\mathbf{E}(t) = \Theta(t) - v_j$ is in P'. □

Remark. *A geometric interpretation of P' is that it is the polytope P enlarged q times and translated so that it includes the origin. To minimize the bound on $\|\mathbf{E}(t)\|$, p_0 should be picked such that the maximum norm of the points in P' is minimized.*

By considering the vectors $\mathbf{E}(t)$ and $\Theta(t)$ in Section 6 as points in $\mathbb{R}^n$, Theorem 7.2 can be used to give an upper bound for $\mathbf{B}$ on the order of $\Omega(dn)$ for the problem in Section 6 where we set $q = d$.

8. Multiple Polytopes

In this section we generalize the problem in Section 7 to multiple polytopes. We assume that there are given a finite number of polytopes $P_1, \dots P_r$. At each time t, a polytope $P_{m_t}, m_t = 1, \dots r$ is chosen and the demand $\mathbf{\Lambda}(t)$ at time t is a point in P_{m_t}. The vertex chosen at time t must be a vertex of the polytope P_{m_t}. We consider here the protocol that allows errors to be treated by groups corresponding to the polytopes that generate them, to the contrary of what was considered in [13] where (in a protocol than contains the GCP as a special case) errors were cumulative across all the polytopes $P_1, \dots P_r$ and the algorithm was fixed to be greedy. The problem that we consider here is a sort of product of individual one polytope problems and can be reduced to the single polytope case as follows:

Theorem 8.1. *For each k, let S_k be a set such that the geometric problem with a single polytope P_k has its error within S_k for all time, then the error $\mathbf{E}(t)$ for the multiple polytope case satisfies: $\mathbf{E}(t) \in \sum_{k=1}^{m} S_k$.*

Proof. The theorem can be proved by decomposing $\mathbf{E}(t)$ into m components, one for each polytope: $\mathbf{E}(t) = \mathbf{E}_1(t) + \cdots + \mathbf{E}_m(t)$. At each time t the polytope P_{m_t} is chosen, and the error component $\mathbf{E}_{m_t}(t)$ is updated according to the geometric problem for a single polytope P_{m_t} while the other components remain the same. □

Remark. *To the contrary of what has been described in Theorem 8.1 about errors for general online algorithms, there is no nice dynamics (generalizing equation* (1.4)*) for the greedy algorithm in the affine space of minimal dimension that would contain more than one polytope. It is shown in* [13] *that any invariant region for the greedy algorithm in the invariant space $\mathcal{A}^n$ of minimal dimension would need to be unbounded for more than one polytope, to the contrary of what is proven in* [1] *for the case of a single polytope.*

Note that in applying Theorem 8.1, polytopes which differs from each other only by a translation can be considered the same: such a simplification would not be possible for the greedy algorithm in affine space given by the obvious generalization of equation (1.4).

When the number of distinct vertices of the set of polytopes is much smaller than the number of polytopes, for instance, when $\{P_m\}$ are the set of faces of a polytope, then the GCP can provide a better upper bound for this problem.

Theorem 8.2. *There exists an on-line algorithm such that:*

$$\|\mathbf{E}(t)\| \le \left(\frac{n-1}{2}\right) \sup_{\sum b_j=0, |b_j|\le 1} \left\| \sum b_j v_j \right\| , \tag{8.1}$$

where $\{v_j\}$ is the set of all the distinct vertices of the set of polytopes $P_1, \ldots, P_m$.

Proof. The proof is essentially the same as Theorem 7.1. The demand vector at each time can be written as a convex combination of the vertices v_j. The restriction that the chosen vertex is a vertex in P_{m_t} corresponds to the fact that the chosen task in GCP must have nonzero demand. In fact, it is weaker since the chosen task having nonzero demand implies that the corresponding vertex is a vertex in P_{m_t} but not vice versa. Thus the bound in Theorem 4.1 can be used for this problem. □

Remark. *The relative positions between the various polytopes do not affect $\mathbf{E}(t)$. Therefore, in order to minimize the bound on $\|\mathbf{E}(t)\|$, the polytopes (at least those without common vertices) should be translated such that the right hand side of equation* (8.1) *is minimized.*

The next theorem gives an explicit construction which shows that this problem has a bounded error under the greedy algorithm for the Euclidean metric on the two dimensional plane.

Theorem 8.3. *$\mathbf{E}(t)$ is bounded under the greedy algorithm for the Euclidean metric in $\mathbb{R}^2$.*

Proof. First we construct a convex polytope P' such that the modified input $\Theta(t)$ is a point in P' for all t. The theorem is then proved by noting that $\mathbf{E}(t) = \Theta(t) - \mathbf{\Lambda}(t)$.

Let n_j^m, $j = 1, \ldots, n_m$ be the unit normal vectors to the edges of P_m. Let P_m^t denote the polygon generated by moving the edges of P_m perpendicularly outward a distance t:

$$P_m^t = \bigcap_j \{x : x \cdot n_j^m \le d_j^m + t\} ,$$

where $P_m = P_m^0$. We then define $P' = \bigcap_m P_m^t$ for large enough t. Consider time t where the polytope is P_{m_t}. By Theorem 2 in [3], for large enough t, $\Theta(t+1) \in P_{m_t}^t$. Next we need to show that $\Theta(t+1) \in P_m^t$ for $m \ne m_t$. Let v_1 be the vertex in P_{m_t} such that $\Theta(t) \in R_{v_1}$. Let n_0 and n_1 be the two adjacent unit normal vectors of v_1 in P_{m_t}. It's clear that (n_0, n_1) form a basis of $\mathbb{R}^2$. Let n_k^m be a normal vector of P_m, $m \neq m_t$. It's clear that n_k^m can be written as $an_0 + bn_1$. If either $a < 0$ or $b < 0$, then the proof of Theorem 2 in [3] can be used to show that $\Theta(n+1) \cdot n_k^m \leq d_k^m + t$ when t is large enough. Assume that both a and b are nonnegative. By induction on t, we assume that $\Theta(t) \cdot n_k^m \leq d_k^m + t$. It then follows that $\Theta(t) \cdot n_k^m \leq d_k^m + t + (\mathbf{\Lambda}(t) - v_1) \cdot n_k^m = d_k^m + t + (\mathbf{\Lambda}(t) - v_1) \cdot (an_0 + bn_1)$. It's easy to see that $(\mathbf{\Lambda}(t) - v_1) \cdot n_0 \leq 0$ and $(\mathbf{\Lambda}(t) - v_1) \cdot n_1 \leq 0$ and thus $\Theta(t+1) \cdot n_k^m \leq d_k^m + t$. $\square$

Remark. *The problem in Section 7 can be thought of as an approximation algorithm: a series of points in P is approximated by a series of vertices of P such that on average the difference is small. Similar considerations apply to the problem in section 8. Consider now the case where instead of vertices we pick any set of points V such that the convex hull of V includes P. Can the error $\mathbf{E}(t)$ be made smaller by choosing V which covers P more densely? Theorems 7.2 and 8.1 can be used to show that if each point in P is inside some the set S which is formed as the convex hull of some points in V such that all these sets S are these same except for a translation, then the error $\mathbf{E}(t)$ is bounded by S expanded n times. Thus if we pick V such that S is small enough, then the error $\mathbf{E}(t)$ will also be small.*

Appendix A. Rate at which $\|E(t)\|_\infty$ approach $\frac{n-1}{2}$

The following result shows the rate at which $\|\mathbf{E}(t)\|_\infty$ approaches $\frac{n-1}{2}$ for the greedy algorithm in Theorem 4.2.

Theorem A.1. *For $n > 2$, it takes $\Omega(n \log(1/\epsilon))$ steps for the error in the greedy algorithm to reach $(1-\epsilon)(N-1)/2$.*

Proof. At time t let $(E_1, \ldots, E_n) = \mathbf{E}(t)$ and $(w_1, \ldots, w_n) = \mathbf{E}(t-1)$. Let ϵ be the smallest value such that for any $k = 1, \ldots, n-1$, and any group of k tasks, we have $E_1 + \cdots + E_k \geq -(1-\epsilon)k(n-k)/2$. Let δ be the smallest value such that for any $k = 1, \ldots, n-1$, any group of k tasks, we have $w_1 + ... + w_k \geq -(1-\delta)k(n-k)/2$.

Both equations hold trivially for $k = 0$ or $k = n$ but ϵ and δ are defined by minimizing over $k = 1, \ldots, n-1$.

Initially $\delta = 1$. We want to bound how quickly ϵ can approach 0. Suppose n_0 steps are required to achieve $\delta < \frac{1}{3}$. Assume throughout now that $\delta < \frac{1}{3}$.

Let $k+1$ be some value between 1 and $n-1$ achieving the new minimum ϵ, *i.e.,* $E_1 + \cdots + E_k + E_{k+1} = (1-\epsilon)(k+1)(N-k-1)/2$. Assume $\epsilon < \delta$ (since otherwise we're very happy). Then $w_1 + ... + w_k + w_{k+1} \leq (1-\delta)(k+1)(N-k-1)/2 < (1-\epsilon)(k+1)(N-k-1)/2$, so some of these quantities grew, implying two things: (1) The chosen task in the last time was among these $k+1$ tasks; call that task $\mathbb{T}_{k+1}$. (2) At least one non chosen task with nonzero demand was among the other $n-k-1$ tasks; call them $\mathbb{T}_{k+2}, \ldots, \mathbb{T}_{k+M}$, with $M \geq 2$ (so $M-1 \geq 1$). Define $P = \lambda_{k+1} + \cdots + \lambda_{k+M}$ to be the demand contributed by the chosen task and those tasks with nonzero demands among the "other" $n-k-1$ tasks. Define $Q = \lambda_1 + \cdots + \lambda_k$ to be the demand contributed by those non chosen tasks with nonzero demands among the first k tasks. We have $P + Q = 1$.

$$\begin{array}{rcl} \sum_{i=1}^{k} w_i & \leq & (1-\delta)(k)(N-k)/2 \\ \sum_{i=1}^{k+M} w_i & \leq & (1-\delta)(k+M)(N-k-M)/2 \\ & = & (1-\delta)[k(N-k) + M(N-2k-M)]/2 \end{array} \quad . \tag{A.1}$$

Adding $(M-1)/M$ times the first equation to $1/M$ times the second equation: $w_1+\cdots+w_k+(1/M)(w_{k+1}+\cdots+w_{k+M}) \le (1-\delta)[k(N-k)+(N-2k-M)]/2$ $x_{k+1}-1 = w_{k+1}-c_{k+1} \le w_{k+t}-c_{k+t}$, $t=1,2,\ldots,M$ $M(x_{k+1}-1) \le w_{k+1}+w_{k+2}+\cdots+w_{k+M}-c_{k+1}-\cdots-c_{k+M} = w_{k+1}+\cdots+w_{k+M}-P$.

$$\text{(A.2)}\quad \begin{array}{rcl} \sum_{i=1}^{k+1} x_i &=& \sum_{i=1}^{k} w_i - \sum_{i=1}^{k} c_i + (x_{k+1}-1)+1 \\ &\le& \sum_{i=1}^{k} w_i - Q + (1/M)\left(\sum_{i=k+1}^{k+M} w_i - P\right)+1 \\ &\le& (1-\delta)[k(N-k)+(N-2k-M)]/2 \\ && -Q-P/M+1 \\ &\le& (1-\delta)[(k+1)(N-k-1)-(M-1)]/2 \\ && +P-P/M \\ &\le& (1-\delta)[(k+1)(N-k-1)]/2 \\ && -(1-\delta)(M-1)/2+1-1/M \end{array}.$$

If $M=2$ we have $-(M-1)/2+1-1/M=0$, so that we would get:

$$\text{(A.3)}\quad \begin{array}{rcl} \sum_{i=1}^{k+1} x_i &\le& (1-\delta)[(k+1)(N-k-1)]/2 \\ && +\delta(M-1)/2 \\ (1-\epsilon)\frac{(k+1)(N-k-1)}{2} &\le& (1-\delta)\frac{(k+1)(N-k-1)}{2}/2 \\ && +\frac{\delta}{2} \\ \epsilon &\ge& \delta\frac{(k+1)(N-k-1)-1}{(k+1)(N-k-1)} \end{array}.$$

In the worst case $(k+1=1)$ we have $\epsilon \ge \delta(1-1/(N-1))$

If $M>2$ we have $-(1-\delta)(M-1)/2+(1-1/M)<0$ because $\delta<1/3$. So in this case we get $\epsilon \ge \delta$.

Thus after $\delta \le 1/3$, each step decreases δ by at most a factor of $(1-1/(N-1))$. So it takes $\Omega(N\log(1/\epsilon))$ to reach a small value of ϵ. □

Appendix B. Details of Theorem 5.1

We list transitions that increase the second moment, starting from any given sequence of small gaps. The list of tasks with nonzero demand is obvious from inspection, and is determined by an adversary; we have listed the result of the greedy algorithm, but in each case one can check that departure from the greedy algorithm does not avoid increase in the second moment.

The sequence of gaps is listed first, then the initial configuration and the transitions to the final configurations, where we abbreviate (a,b,c) for $(E_t+a/6, E_t+b/6, E_t+c/6)$.

GAPS Transitions

($\frac{0}{6}$): $(0,0)\to(-3,3)$

($\frac{1}{6}$): $(0,1)\to(3,-2)$

($\frac{2}{6}$): $(0,2)\to(3,-1)$

($\frac{3}{6},\frac{3}{6}$): $(0,3,6)\to(2,5,2)\to(-1,5,5)\to(-1,2,8)$

($\frac{3}{6},\frac{4}{6}$): $(0,3,7)\to(2,5,3)\to(2,2,6)\to(-1,5,6)\to(-1,8,3)$

($\frac{4}{6},\frac{3}{6}$): $(0,4,7)\to(2,6,3)\to(4,2,5)\to(1,5,5)\to(1,2,8)\to(4,-1,8)$

($\frac{3}{6},\frac{5}{6}$): $(0,3,8)\to(2,5,4)\to(2,2,7)\to(-1,5,7)\to(-1,8,4)$

($\frac{5}{6},\frac{3}{6}$): $(0,5,8)\to(2,7,4)\to(4,3,6)\to(1,6,6)\to(1,3,9)\to(4,0,9)$

($\frac{3}{6},\frac{6}{6},\frac{3}{6}$): $(0,3,9,12)\to(2,5,5,12)\to(2,2,8,12)\to(-1,5,8,12)\to(-1,7,10,8)\to(-1,7,7,11)\to(-1,4,10,11)\to(-1,4,13,8)$

$\mathbf{(\frac{3}{6},\frac{6}{6},\frac{4}{6})}$**:** $(0,3,9,13)\to(2,5,5,13)\to(2,2,8,13)\to(-1,5,8,13)\to(-1,7,10,9)\to(-1,7,7,12)\to(-1,4,10,12)\to(-1,4,13,9)$

$\mathbf{(\frac{4}{6},\frac{6}{6},\frac{3}{6})}$**:** $(0,4,10,13)\to(0,6,12,9)\to(0,8,8,11)\to(0,5,11,11)\to(0,5,8,14)\to(2,7,4,14)\to(4,3,6,14)\to(1,6,6,14)\to(1,3,9,14)\to(4,0,9,14)$

$\mathbf{(\frac{4}{6},\frac{4}{6},\frac{4}{6})}$**:** $(0,4,8,12)\to(0,4,11,9)\to(0,7,11,6)\to(2,3,11,8)\to(5,0,11,8)\to(8,0,8,8)\to(4,0,10,10)\to(4,0,7,13)\to(4,3,4,13)\to(6,5,0,13)\to(2,7,2,13)\to(-1,7,5,13)\to(-1,4,8,13)$

$\mathbf{(\frac{4}{6},\frac{4}{6},\frac{5}{6})}$**:** $(0,4,8,13)\to(3,1,8,13)\to(3,3,10,9)\to(5,5,10,5)\to(7,7,10,1)\to(4,10,10,1)\to(4,7,13,1)\to(4,4,13,4)\to(6,6,13,0)\to(8,2,13,2)\to(11,2,10,2)\to(11,5,10,-1)\to(7,7,12,-1)\to(4,10,12,-1)\to(4,13,9,-1)$

$\mathbf{(\frac{5}{6},\frac{4}{6},\frac{4}{6})}$**:** $(0,5,9,13)\to(0,5,12,10)\to(2,5,8,12)\to(5,5,5,12)\to(1,7,7,12)\to(3,3,9,12)\to(0,6,9,12)\to(0,9,9,9)\to(0,5,11,11)\to(0,5,8,14)\to(3,5,5,14)\to(5,1,7,14)\to(7,3,3,14)\to(7,0,6,14)\to(4,0,9,14)$

References

[1] R. Adler, B. Kitchens, M. Martens, C. Pugh, M. Shub, and C. Tresser. Convex dynamics and applications. *Ergod. Th. & Dynam. Sys.*, **25**, 321–352, 2005.

[2] R. Adler, B. Kitchens, M. Martens, A. Nogueira, C. Tresser, and C. W. Wu. Error bounds for error diffusion and related digital halftoning algorithms. In *Proceedings IEEE International Symposium on Circuits and Systems, Volume* **II**, II 513–II 516, 2001.

[3] R. L. Adler, B. P. Kitchens, M. Martens, C. P. Tresser, and C. W. Wu. The mathematics of halftoning. *IBM Journal of Research and Development*, **47**(1), 5–15, 2003.

[4] M. Ajtai, J. Aspnes, M. Naor, Y. Rabani, L. J. Schulman, and O. Waarts. Fairness in scheduling. *Journal of Algorithms*, **29**(2), 306–357, 1998.

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

[6] R. Fagin and J. H. Williams. A fair carpool scheduling algorithm. *IBM Journal of Research and Development*, **27**(2),133–139, 1983.

[7] D. E. Knuth *The Art of Computer Programming.* Addison Wesley, 1968.

[8] H. G. Meijer. On a distribution problem in finite sets. *Nederl. Akad. Wetersch. Indag. Math.*, **35**, 9–17, 1973.

[9] H. Niederreiter. On the existence of uniformly distributed sequences in compact spaces. *Compositio Math.*, **25**, 93–99, 1972.

[10] H. Nowicki and C. Tresser. Convex dynamics: properties of invariant sets. *Nonlinearity.*, **17**, 1645–1676, 2004.

[11] R. Tijdeman. On a distribution problem in finite and countable sets. *Journal of Combinatorial Theory (A)*, **15**, 129–137, 1973.

[12] R. Tijdeman. The chairman assignment problem. *Discrete Mathematics*, **32**, 323–330, 1980.

[13] C. Tresser. Dynamique de la diffusion de l'erreur sur plusieurs polytopes. *Comptes Rendus Mathematique*, **338**, 793–798, 2004.

IBM, P. O. Box 218, Yorktown Heights, NY 10598, U.S.A.