Fairness over Time in Dynamic Resource Allocation with an Application in Healthcare

Andrea Lodi[1,2], Philippe Olivier[1,2], Gilles Pesant[2], and Sriram Sankaranarayanan*[3]

[1] *CERC*
[2] *Polytechnique Montréal, Canada*
[3] *IIM Ahmedabad, India*

Abstract

Decision making problems are typically concerned with maximizing efficiency. In contrast, we address problems where there are multiple stakeholders and a centralized decision maker who is obliged to decide in a fair manner. Different decisions give different utility to each stakeholder. In cases where these decisions are made repeatedly, we provide efficient mathematical programming formulations to identify both the maximum fairness possible and the decisions that improve fairness over time, for reasonable metrics of fairness. We apply this framework to the problem of ambulance allocation, where decisions in consecutive rounds are constrained. With this additional complexity, we prove structural results on identifying fair feasible allocation policies and provide a hybrid algorithm with column generation and constraint programming-based solution techniques for this class of problems. Computational experiments show that our method can solve these problems orders of magnitude faster than a naive approach.

1 Introduction

A resource allocation problem can be defined as the problem of choosing an allocation from a set of feasible allocations to a finite set of stakeholders to minimize some objective function.

Generally, this objective function represents the efficiency of an allocation by considering, for example, the total cost incurred by all the stakeholders (Toshihide Ibaraki, 1988).

However, in some cases, such an objective may be inappropriate due to ethical or moral considerations. For instance, to whom should limited medical supplies go in a time of crisis (Heier Stamm et al., 2017)? These situations call for a different paradigm of decision-making that addresses the need of *fairness* in resource allocation.

Resource allocation problems trace their roots to the 1950s when the division of effort between two tasks was studied (Koopman, 1953). Fairness in resource allocation problems has been studied since the 1960s. A mathematical model for the fair apportionment of the U.S. House of Representatives considered the minimization of the difference between the disparity of representation between any pairwise states (Burt and Harris, 1963). This problem was revisited about two decades later (Katoh et al., 1985, Zeitlin, 1981). The so-called *minimax* and *maximin* objectives in resource allocation, which achieve some fairness by either minimizing the maximum (or maximizing the minimum) number of resources allocated to entities, have been studied at least since the 1970s (Jacobsen, 1971, Porteus and Yormark, 1972). A recent book on equitable resource allocation presents various models and advocates a lexicographic minimax/maximin approach for fair and efficient solutions (Luss, 2012). This work further discusses multiperiod equitable resource allocation. Fair resource allocation over time is also considered in the dynamic case where resource availability changes over time and is solved using approximation algorithms (Bampis et al., 2018). A combination of

*srirams@iima.ac.in

efficiency and fairness in resource allocation is common in communication networks (Ogryczak et al., 2014, Ogryczak, Wlodzimierz, 2014), and can be found in various other fields, such as in some pickup and delivery problems (Eisenhandler and Tzur, 2019).

Our Contributions. Our first contribution is to formally set up an abstract framework for repeatedly solving a fair allocation problem such that enhanced fairness is achieved over time, as opposed to a single round of allocation. Typically, a fair allocation could turn out to be an inefficient allocation. The framework addresses this trade-off by explicitly providing adequate control to the decision-maker on the level of the desired efficiency of the solutions.

Then, we prove that the concept of achieving fairness over time is not useful should the set of feasible allocations be a convex set, as long as the value each stakeholder obtained is a linear function of the allocation. We prove this result irrespective of the measure of fairness one uses, as long as the measure has crucial technical properties mentioned formally later. Next, we show closed-form and tight solutions for the number of rounds required for perfectly fair solutions, for special choices of the set of allocations that could be of practical interest.

However, there might also be other cases of practical interest where such closed-form solutions are harder, if not impossible, to obtain. We provide an integer programming formulation to solve these problems. The formulation hence provided can be combined directly with delayed column generation techniques in case of large instances.

With the above results and ideas, we consider the problem of allocating ambulances to multiple areas of a city, where the residents of each area are the stakeholders. This family of problems is easier than the general case as there exists an efficient greedy algorithm that provides reasonable solutions fast. However, such solutions could be far from optimal, and we provide examples where the greedy algorithm fails. Besides that, in this family of problems, we also impose restrictions on how allocations in successive rounds could be. This is reminiscent of the real-life restriction that one does not want to relocate too many ambulance vehicles each day. This added constraint makes identifying good solutions much harder as the proposed column generation-technique becomes invalid now. To solve this problem, we identify a graph induced by any candidate solution and show that deciding the feasibility of a candidate solution provided by column generation is equivalent to identifying Hamiltonian paths in this graph. Hence, we provide multiple subroutines to retain the validity of the column generation-based approach. We also provide experimental results on the computational efficiency of our method.

The paper is organized as follows. Section 2 introduces some basic definitions and concepts. Section 3 presents some theoretical proofs for simple cases, and Section 4 establishes a general framework which allows to tackle more complicated cases. This framework is used to solve a practical problem in Section 5, whose results are presented in Section 6. Finally, some conclusions are drawn in Section 7.

2 Preliminaries

In this section, we formally define the terms used throughout the manuscript. We study resource allocation in the context of fairness over time, meaning that resources are allocated to stakeholders over some time horizon. We use $\mathcal{X}$ to denote the set of all feasible allocations, with $n \in \mathbb{Z}_{\geq 0}$, $n \geq 2$ being the number of stakeholders among whom the allocations should be done in a fair way.

Definition 1 (Benefit function)**.** *The benefit function $\tau : \mathcal{X} \to \mathbb{R}^n$ is a function such that the i-th component $[\tau(x)]_i$ refers to the benefit or utility enjoyed by the i-th stakeholder due to the allocation $x \in \mathcal{X}$.*

Assume ϕ to be a function that measures the unfairness associated with a given benefit pattern for the n stakeholders—a concept formalized in Definition 2—which is to be minimized. In this context, a basic fair resource allocation problem can be defined simply as

$$\min_{x \in \mathcal{X}} \quad \phi(\tau(x)). \tag{1}$$

Many decision-makers are public servants who are either elected or nominated, and who serve in their position for a predetermined amount of time. In the context of serving the public, these officials must often juggle with limited means to ensure that everyone they serve is catered for. Public satisfaction naturally plays a role in the evaluation of their performance. Herein lies the motivation to solve resource allocation problems that consider fairness over time.

Assuming that there are T rounds of decision making, model (1) can be expanded to

$$\min_{x(t),y} \quad \phi(y) \tag{2a}$$

$$\text{s.t.} \quad y = \frac{1}{T}\sum_{t=1}^{T} \tau(x(t)) \tag{2b}$$

$$x(t) \in \mathcal{X}, \qquad \forall\, i = 1, \ldots, T. \tag{2c}$$

Here, we implicitly assume that it is sensible to add the benefits obtained at different rounds of decision making, and use the average benefit over time to compute fairness.

Now, in this section, we first define the properties that a general measure of unfairness should have for it to be considered valid in our context. To start with, we state that (i) if all the stakeholders of the problem get the same benefit or utility, then the unfairness associated with such an allocation of benefits should be 0, and (ii) the unfairness associated with benefits should be invariant to permutations of the ordering of stakeholders.

Definition 2 (Unfairness function)**.** *Given $y \in \mathbb{R}^n$, $\phi : \mathbb{R}^n \to \mathbb{R}_{\geq 0}$ determines the extent of unfairness in the allocations, if the i-th stakeholder gets a benefit of y_i. Such a function ϕ satisfies the following:*

1. $\phi(y_1, \ldots, y_n) = 0 \iff y_1 = y_2 = \ldots = y_n$,
2. $\phi(y_1, \ldots, y_n) = \phi(\pi_1(y), \ldots, \pi_n(y))$ *for any permutation π.*

In addition, if ϕ is a convex function, we call ϕ a convex unfairness function*.*

In general, trivial solutions where no benefit is obtained by any of the stakeholders, i.e., when $\tau(x)$ is a zero vector, are perfectly fair solutions! However, this results in a gross loss of efficiency.

Definition 3 (Inefficiency function)**.** *Given $\mathcal{X}$ and the benefit function τ, the inefficiency function $\eta : \mathcal{X} \to [0, 1]$ is defined as*

$$\eta(x) \quad = \quad \begin{cases} 0 & \text{if } \overline{f} = \underline{f}, \\ \frac{\overline{f} - \sum_{i=1}^{n} [\tau(x)]_i}{\overline{f} - \underline{f}} & \text{otherwise,} \end{cases} \tag{3}$$

where $\overline{f} = \sup_{x \in \mathcal{X}} \sum_{i=1}^{n} [\tau(x)]_i$, $\underline{f} = \inf_{x \in \mathcal{X}} \sum_{i=1}^{n} [\tau(x)]_i$, and $\overline{f}$ and $\underline{f}$ are assumed to be finite.

Remark 4. Note that for all feasible $x \in \mathcal{X}$, we indeed have $\eta(x) \in [0, 1]$. For the most efficient x, i.e., the x (or allocation) that maximizes the sum of benefits, $\eta(x) = 0$, while for the least efficient x, we have $\eta(x) = 1$. Thus, η serves as a method of normalization of the objective values.

Definition 5. *Equivalence classes defined due to the function η and partitioning $\mathcal{X}$ are, for any $\overline{\eta} \in [0, 1]$*

$$\widetilde{\mathcal{X}}_{\overline{\eta}} \quad := \quad \{x \in \mathcal{X} : \eta(x) = \overline{\eta}\}. \tag{4}$$

The above definition enables to succinctly represent a target to find fair solutions *only* if they are also sufficiently efficient. For instance, one might restrict the feasible set to $\cup_{\eta : \eta < \overline{\eta}} \widetilde{\mathcal{X}}_{\eta}$.

We now define a single-period fair allocation problem, subject to efficiency constraints. One might always choose $\eta = 1$ to retrieve the problem in model (1) without efficiency constraints.

Definition 6 (Single-period fair allocation (SPFA) problem)**.** *Given $\overline{\eta} \in [0,1]$, the single-period fair allocation problem is to solve*

$$\min_{x} \quad \phi(\tau(x)) \tag{5a}$$

$$s.t. \quad x \in \bigcup_{\eta:\eta\in[0,\overline{\eta}]} \widetilde{\mathcal{X}}_\eta. \tag{5b}$$

Example 7 motivates and provides a mean of validation for model (5). It shows that with reasonable choices, we retrieve the intuitively fair solution.

Example 7. *Consider the case where $\mathcal{X} = \{x \in \mathbb{R}^n_+ : \sum_{i=1}^n x_i = 1\}$, i.e., $\mathcal{X}$ is a simplex. Further, assume that $[\tau(x)]_i = \tau_i\, x_i$ for some scalars $\tau_i \gneq 0$, and that, w.l.o.g., $\tau_1 \geq \tau_2 \geq \ldots \geq \tau_n$.*

For the choice of $\overline{\eta} = 1$, i.e., with no efficiency constraints, the solution for the SPFA problem is given by

$$x_i^\star = \frac{g}{\tau_i}, \quad \forall i \tag{6a}$$

$$where \quad g = \frac{1}{\sum_{i=1}^n \frac{1}{\tau_i}}. \tag{6b}$$

Clearly, each stakeholder enjoys a benefit of g, and hence the unfairness associated with this allocation is 0. We can notice that $\eta(x^\star) = \frac{\tau_1 - ng}{\tau_1 - \tau_n}$.

Note that for the case where $n = 2$, the above simplifies to

$$\begin{aligned} x_1^\star &= \frac{\tau_2}{\tau_1+\tau_2} \\ x_2^\star &= \frac{\tau_1}{\tau_1+\tau_2} \\ \eta(x^\star) &= \frac{\tau_1^2 - \tau_1\tau_2}{\tau_1^2 - \tau_2^2}. \end{aligned}$$

We now define the fairness-over-time problem.

Definition 8 (T-period fair allocation (T-PFA) problem)**.** *Given $\overline{\eta} \in [0,1]$ and $T \in \mathbb{Z}_{\geq 0}$ with $T \geq 2$, the T-period fair allocation problem is to solve*

$$\min_{x(t),y} \quad \phi(y) \tag{7a}$$

$$s.t. \quad y = \frac{1}{T}\sum_{t=1}^{T} \tau(x(t)) \tag{7b}$$

$$x(t) \in \bigcup_{\eta:\eta\in[0,\overline{\eta}]} \widetilde{\mathcal{X}}_\eta, \quad \forall\, i = 1,\ldots,T. \tag{7c}$$

We say that a fair-allocation problem (SPFA or T-PFA) has *perfect fairness* if the optimal objective value of the corresponding optimization problems is 0.

Example 9 (Usefulness of the T-PFA)**.** *Let $\mathcal{X} = \{x \in \{0,1\}^2 : x_1 + x_2 = 1\}$. Further, let $\tau(x) = (2x_1, x_2)$. Let $\overline{\eta} = 1$. Note that, in the case of the SPFA, the optimal objective is necessarily nonzero, since the benefits of the two stakeholders are unequal for every feasible solution. However, consider the 3-PFA. Now, if $x(1) = (1,0)$, $x(2) = (0,1)$, $x(3) = (0,1)$, $y = \left(\frac{2}{3}, \frac{2}{3}\right)$. So, for any choice of ϕ, the optimal objective value is 0, which is strictly better than the SPFA solution.*

3 Simple Cases

3.1 Convex $\mathcal{X}$

We have motivated in Example 9 that ensuring fairness over multiple rounds could offer better results than seeking fairness on a per-round basis. We now show that this holds only for a nonconvex $\mathcal{X}$. In other words, if $\mathcal{X}$ is convex, we necessarily get no improvement in T-PFA over SPFA.

Theorem 10. *Let $\mathcal{X}$ be convex. Further, let τ be a linear function. Let $\overline{\eta}$ be given. Let $f^\star$ be the optimal value of the SPFA problem and let $f_2^\star, f_3^\star, \ldots$ be the optimal values of the T-PFA problem with $T = 2, 3, \ldots$. Then $f^\star = f_2^\star = f_3^\star \ldots$.*

Proof. Given that τ is a linear function, we can write $[\tau(x)]_i = \tau_i^\mathsf{T} x$ for an appropriately chosen τ_i, for $i = 1, \ldots, n$. Suppose that $x^\star(t)$, for $t = 1, \ldots, T$, and $y^\star$ solve the T-PFA problem. By our notation, this has an objective value of $\phi(y^\star) = f_T^\star$. Now, we construct a solution for the SPFA problem with an objective value equal to $f_T^\star$. For this, consider the point $\overline{x} = \frac{1}{T}\sum_{t=1}^{T} x^\star(t)$. First we claim that $\overline{x} \in \mathcal{X}$.

We have that $x(t) \in \widehat{\mathcal{X}}$ where $\widehat{\mathcal{X}} = \cup_{\eta:\eta\in[0,\overline{\eta}]}\widetilde{\mathcal{X}}_\eta$. Now, we observe that $\widehat{\mathcal{X}}$ is a convex set given by $\{x \in \mathcal{X} : (\sum_{i=1}^n \tau_i)^\mathsf{T} x \geq \overline{f} - (\overline{f} - \underline{f})\overline{\eta}\}$, where $\overline{f}$ and $\underline{f}$ are constants. Since $\overline{x}$ is obtained as a convex combination of $x^\star(t) \in \widehat{\mathcal{X}}$, it follows that $\overline{x} \in \widehat{\mathcal{X}}$. Finally, from the linearity of τ, we can set $\overline{y} = y^\star$. This shows that $(\overline{x}, \overline{y})$ is feasible. Since $\overline{y} = y^\star$, it follows that their objective function values are equal. ■ □

Having proven that multiple rounds of allocation cannot improve the fairness when $\mathcal{X}$ is convex, we show that it is not necessarily due to the fact that perfect fairness is obtainable in SPFA. Example 11 shows an instance where the unfairness is strictly positive with a single round of allocation, irrespective of the choice of ϕ. Naturally, due to Theorem 10, perfect fairness is neither possible with multiple rounds of allocation.

Example 11. *Consider a fair allocation problem where $\mathcal{X} = \{x \in \mathbb{R}^3_{\geq 0} : x_1 + x_2 + x_3 = 1\}$. Clearly $\mathcal{X}$ is convex. Consider linear τ defined as*

$$\begin{aligned} [\tau(x)]_1 &= x_1 + \frac{3}{4}x_2 + \frac{3}{4}x_3 \\ [\tau(x)]_2 &= x_2 \\ [\tau(x)]_3 &= x_3. \end{aligned}$$

Let $\overline{x}$ be a fair allocation. In such a case, we need $[\tau(\overline{x})]_1 = [\tau(\overline{x})]_2 = [\tau(\overline{x})]_3$. Thus, we need

$$\begin{aligned} \rho &= x_1 + \frac{3}{4}x_2 + \frac{3}{4}x_3 \\ \rho &= x_2 \\ \rho &= x_3 \end{aligned}$$

for some $\rho \in \mathbb{R}_+$. Solving this linear system gives the unique *fair allocation as allocations of the form $(x_1, x_2, x_3) = (-0.5\rho, \rho, \rho)$, which necessarily violates the non-negativity constraints in the definition of $\mathcal{X}$. Any other allocation necessarily has $\phi(\tau(x)) > 0$.*

3.2 Simplicial $\mathcal{X}$

The first part of Theorem 12 states that for any n and a $\mathcal{X}$ of the specified form, perfect fairness is possible precisely after $\overline{T}$ periods, provided the efficiency requirements are as stated. The second part shows tightness of the results saying, for any other stricter efficiency requirements, there exists a counterexample with just two stakeholders, such that perfect fairness is impossible in $\overline{T}$ steps.

Theorem 12. *Let $\mathcal{X} = \{x \in \mathbb{Z}^n_{\geq 0} : \sum_{i=1}^n x_i \leq a\}$ for some positive integer a. Let the benefit function be $[\tau(x)]_i = \tau_i x_i$, where each τ_i is a positive integer. Assume, w.l.o.g., that $\tau_1 \geq \tau_2 \geq \ldots \geq \tau_n > 0$. Let $L = LCM(\tau_1, \ldots, \tau_n)$. Then,*

1. *Perfect fairness cannot be obtained in T periods where T is strictly lesser than $\overline{T} = \left\lceil \frac{1}{a} \sum_{i=1}^{n} \frac{L}{\tau_i} \right\rceil$. However, if $1 > \overline{\eta} \geq \frac{\tau_1 - \min(\tau_n, \tau_1/a)}{\tau_1}$, and if $\overline{T} > 1$, perfect fairness can be achieved in exactly $\overline{T}$ periods.*

2. *Given any $n \geq 2$, there exist $\tau_1 \geq \tau_2 \geq \ldots \geq \tau_n$ and $a \in \mathbb{Z}_{>0}$ such that perfect fairness is unattainable for T-PFA where $T \leq \overline{T}$ as above with $\overline{\eta} < \frac{\tau_1 - \min(\tau_n, \tau_1/a)}{\tau_1}$.*

Proof. For the first part, observe that an unfairness of 0 can be achieved if and only if the total benefit over time for each stakeholder should add up to the same value. Since x_i and τ_i are all integers and we have integer number of periods, the smallest value that the benefits can all add up to is 0, and the next smallest value is L. But the strict inequality $\overline{\eta} < 1$ excludes the trivial allocation of $(0, 0, \ldots, 0)$. A total benefit of L can be achieved for stakeholder i if and only if $\sum_{t=1}^{T} x_i(t) = L/\tau_i$. Summing this over all stakeholders, we get

$$\begin{aligned} \sum_{i=1}^{n} \frac{L}{\tau_i} &= \sum_{i=1}^{n} \sum_{t=1}^{T} x_i(t) \\ &= \sum_{t=1}^{T} \sum_{i=1}^{n} x_i(t) \\ &\leq \sum_{t=1}^{T} a \\ &= aT \\ \implies \quad \frac{1}{a} \sum_{i=1}^{n} \frac{L}{\tau_i} &\leq T \end{aligned}$$

and the ceiling follows from the fact that T is an integer. Now that we have shown that we need at least $\overline{T}$-PFA, we show that the unfairness can be made to 0 in exactly $\overline{T}$ periods. For this, consider a reverse-lexicographic allocation method where all allocations are made to stakeholder n until the total allocation sums to L/τ_n. Next, allocate similarly for $n-1$, $n-2$ and so on up to stakeholder 1. Observe that each of these allocations at any time t is in $\mathcal{X}$ by construction. To show that the allocation in each step has an inefficiency value at most $\overline{\eta}$, consider the two most inefficient allocations. This could happen in the first period, when all allocations are to the stakeholder n, i.e., $x(1) = (0, 0, \ldots, 0, a)$. The inefficiency function value for this allocation is $\frac{a\tau_1 - a\tau_n}{a\tau_1 - 0} = \frac{\tau_1 - \tau_n}{\tau_1} \leq \overline{\eta}$. Alternatively, the most inefficient allocation could be in the last period, where a single allocation is made to the first stakeholder, i.e., $x(T) = (1, 0, 0, \ldots, 0)$. The inefficiency function value for this allocation is $\frac{a\tau_1 - \tau_1}{a\tau_1 - 0} = \frac{\tau_1 - \tau_1/a}{\tau_1} \leq \overline{\eta}$. This proves the first part.

We prove the second part for $n = 2$ first, and then generalize it to $n \geq 3$. Let the target $\overline{T} = \widehat{T} \geq 2$. Choose $\tau_1 = (\widehat{T} - 1)a$ and $\tau_2 = 1$. The LCM $L = (\widehat{T} - 1)a$. From the first part, perfect fairness is not obtainable for $T < \frac{1}{a}\left(\frac{(\widehat{T}-1)a}{(\widehat{T}-1)a} + \frac{(\widehat{T}-1)a}{1}\right) = (\widehat{T} - 1) + \frac{1}{a}$ periods. But from the integrality of the number of periods, perfect fairness is not possible using up to $\widehat{T} - 1$ periods. We now show that perfect fairness is not possible with $\widehat{T}$ periods either. Note that the set of solutions where perfect fairness is achieved in $\widehat{T}$ time steps are all in the form

$$\begin{aligned} x(1) &= (0, a - \delta_1) \\ x(2) &= (0, a - \delta_2) \\ &\vdots \\ x(\widehat{T} - 1) &= (0, a - \delta_{\widehat{T}-1}) \end{aligned}$$

$$x(\widehat{T}) \quad = \quad \left(\sum_{i=1}^{\widehat{T}-1} \delta_i, 1 \right)$$

and its permutations over time for some integers $\delta_i \in \{1, 2, \ldots, a\}$ such that $0 \leq \sum_{i=1}^{\widehat{T}-1} \delta_i \leq a-1$. Now consider the inefficiency function of the allocation $x(1)$; $\eta(x(1)) = \frac{(\widehat{T}-1)a-1+\delta_1/a}{\widehat{T}-1} > \overline{\eta}$ shows that it is infeasible. Thus, any perfectly fair allocation for $\overline{T}$-PFA problem is an infeasible allocation, providing the necessary counterexample. To extend this for $n > 2$, choose $\tau_2 = \tau_3 = \ldots = \tau_{n-1} = 2$, and analogous arguments provide the counterexample. ■ □

Example 13. *Consider Theorem 12 applied to Example 9. We have* $\mathcal{X} = \{(1,0),(0,1)\}$*, and thus* $a = 1$*. Then,* $\tau(x) = (2x_1, x_2)$*, so* $\tau_1 = 2$ *and* $\tau_2 = 1$*, and* $\tau_1 \geq \tau_2 \geq 0$ *holds. In addition,* $L = LCM(2,1) = 2$*. Then,* $\overline{T} = \left\lceil \frac{1}{a} \sum_{i=1}^{n} \frac{L}{\tau_i} \right\rceil = \left\lceil \frac{1}{1}(\frac{2}{2} + \frac{2}{1}) \right\rceil = 3$.

3.3 Sparse simplicial $\mathcal{X}$

The sparse simplicial form for $\mathcal{X}$ is another interesting case from a practitioner's perspective. For example, a government might want to allot money n possible projects, but if it divides among all of them, then it might be insufficient for any of them. So, there could be a restriction that the government allots it to at most r projects.

Theorem 14. *Let* $\mathcal{X} = \{x \in \mathbb{R}^n_{\geq 0} : \sum_{i=1}^{n} x_i \leq a, \|x\|_0 \leq r\}$ *where* $\|\cdot\|_0$ *is the sparsity pseudo-norm, which counts the number of non-zero entries in its argument. Assume that the benefit function is* $[\tau(x)]_i = \tau_i x_i$*, where each* τ_i *is a positive integer. Assume, w.l.o.g., that* $\tau_1 \geq \tau_2 \geq \ldots \geq \tau_n > 0$*. We denote by* (p, q) *the quotient and the remainder for* $\frac{n}{r}$*. Let* $1 > \overline{\eta} \geq \frac{a\,\tau_1 - qv}{a\,\tau_1}$ *where* $v = a \min\left\{\sum_{i=1}^{q} \frac{1}{\tau_i}, \sum_{i=0}^{r-1} \frac{1}{\tau_{n-i}}\right\}$*. Then,* $\overline{T} = \left\lceil \frac{n}{r} \right\rceil$ *is the smallest number of periods required to achieve perfect fairness.*

Proof. Consider the following allocation: In period t for $1 \leq t \leq \overline{T}-1$, allocate $\frac{v}{\tau_i}$ for $i = n-(t-1)r-1, n-(t-1)r-2, \ldots, n-tr$ and 0 to the rest. Note that in each period, we allocate exactly to r stakeholders, and that the inequality constraint is satisfied due to the definition of v and the ordering of $\tau_1 \geq \tau_2 \ldots \geq \tau_n$. We can ensure that in the first $\overline{T}-1$ periods, each of the allocated player receives a value of v, and hence the total benefit of the allocation is rv, giving the $\eta(x(t)) = \frac{a\,\tau_1 - rv}{a\,\tau_1} \leq \overline{\eta}$. In the last period, we allocate $\frac{v}{\tau_i}$ for $i = 1, 2, \ldots, r$. Feasibility follows as before and $\eta(x(\overline{T})) = \frac{a\,\tau_1 - qv}{a\,\tau_1} \leq \overline{\eta}$, which is feasible. Perfect fairness follows since the benefit to each player is invariant.

To prove that the given $\overline{T}$ is the smallest number of periods required to obtain perfect fairness, observe that, for any fewer number of periods, it is impossible for all players to get a non-zero allocation. ■ □

Remark 15. Note that, unlike earlier, we do not have tightness in η now.

Remark 16. We note that the above results, Theorems 10, 12 and 14, are agnostic to the choice of ϕ, and only use the fact that $\phi(x_1, \ldots . x_m) = 0 \iff x_1 = \ldots = x_n$. Any result that holds for $\phi(\cdot) \neq 0$ has to necessarily depend upon the choice of ϕ.

Example 17. *Consider Theorem 14 applied to Example 9. We have* $a = 1$ *and* $r = 2$*. Again,* $\tau(x) = (2x_1, x_2)$*, so* $\tau_1 = 2$ *and* $\tau_2 = 1$*, and* $\tau_1 \geq \tau_2 \geq 0$ *holds.* $\overline{T} = \left\lceil \frac{n}{r} \right\rceil = \left\lceil \frac{2}{2} \right\rceil = 1$*. Indeed, the solution* $(\frac{1}{3}, \frac{2}{3})$ *is feasible and fair.*

4 General Combinatorial $\mathcal{X}$

In many practical areas of interest, $\mathcal{X}$ could be more complicated than the sets presented in Section 3. Let $\mathcal{X} = \{x^1, x^2, \ldots, x^k\}$. We assume that $\mathcal{X}$ is finite, but typically has a large number of elements, given in the form of a solution to a combinatorial problem. We consider a benefit function where $\tau(x) = \Gamma x$, for some

matrix Γ. We thus have $[\tau(x)]_i = \tau_i^\mathsf{T} x$ for $i = 1, \ldots, n$.[1] The efficiency constraint here is trivial, as the inefficient allocations x^j can be removed from $\mathcal{X}$ to retain another (smaller) finite $\mathcal{X}$. In fact, with linear τ, the efficiency constraint is a linear inequality.

Let q_j be the *number of times* the allocation $x^j \in \mathcal{X}$ is chosen over T rounds of decision. In this setting, the T-PFA problem can be restated as[2]

$$\min_{q,y} \quad \phi(y) \tag{8a}$$

$$\text{s.t.} \quad y_i = \tau_i^\mathsf{T} \left(\sum_{j=1}^{k} q_j x^j \right), \quad \forall\, i = 1, \ldots, n \tag{8b}$$

$$\sum_{j=1}^{k} q_j = T \tag{8c}$$

$$q_j \in \mathbb{Z}_{\geq 0}, \qquad \forall\, j = 1, \ldots, k. \tag{8d}$$

Note that since the theorems of Section 3 do not extend to this case, and since Example 11 shows that a perfectly fair solution may not even exist, we know neither the value of the fairest feasible solution, nor the smallest value of T that guarantees such a solution. We now present a two-phase integer program that finds the smallest value of T that guarantees the fairest feasible solution. In the first phase, we are interested in finding the fairest feasible solution. This is achieved by solving model (8), replacing (8c) with $\sum_{j=1}^{k} q_j \geq 1$ to ensure that not only the fairest solution is returned, but that this solution not be $(0, 0, \ldots, 0)$. Let $\phi^\star$ be the value of the solution found in the first phase. In the second phase, we are interested in finding the smallest T which can accommodate a solution of value $\phi^\star$. This is achieved by solving model (8), again replacing (8c) with $\sum_{j=1}^{k} x_j \geq 1$, adding $\phi(y) = \phi^\star$, and changing the objective to $\min_{q,y} \sum_{j=1}^{k} q_j$.

We should note that the value of $T^\star$ found by the second phase could be quite high, even for simple choices of $\mathcal{X}$. If perfect fairness can be attained in a time horizon of size, say, 10000, and that a discrete time point represents a day, then this period of 27-odd years would probably not be suitable for most practical applications, since it is only by reaching the end of the time horizon that optimal fairness is guaranteed. In practice, then, large time horizons can be problematic. This motivates the need for consistent fairness on a smaller scale.

Some compromises can be made which would mitigate this issue. Manually fixing T to a value not higher than the expected duration of the problem would ensure a fair distribution of resources, since reaching the end of the time horizon would be assured. Accepting some degree of unfairness would also decrease the length of the time horizon and achieve similar results. Example 18 illustrates these two options.

Example 18. *Consider a fair allocation problem where* $\mathcal{X} = \{x \in \mathbb{Z}^2_{\geq 0} : x_1 + x_2 = 1\}$. *Further consider* τ *defined as*

$$\begin{aligned} [\tau(x)]_1 &= x_1 + \frac{15}{37} x_2 \\ [\tau(x)]_2 &= x_2 + \frac{15}{47} x_1. \end{aligned}$$

Perfect fairness can be achieved when $T = 1109$, *but if we fix* $T = 5$, *the difference between* y_1 *and* y_2 *will be a mere* ~ 0.42, *and if we allow a small difference of* 0.01 *between* y_1 *and* y_2, *this could be achieved when* $T = 15$.

Another compromise would be to choose a fair way of reaching the solution. Given that we know the value of $T^\star$ which grants optimal fairness, we could then identify the *best path* leading to the optimal solution.

[1] Matrix Γ can implicitly be seen as an adjacency matrix in order to consider a graph representation of the problem.

[2] The reader may have observed that an approach based on column generation would lend itself well for such a model. This is discussed in Section 5.

The best path is the one which ensures that unfairness is kept as low as possible in the intermediate time points, without sacrificing optimality at the end of the time horizon. This would, however, require solving another linear program, which may prove burdensome if the time horizon were too large.

One may think that greedily aiming for the fairest solution at each time point—while considering any unfairness incurred in previous time points, and enforcing minimal efficiency constraints by ensuring that all available resources are used at each time point—could lead to optimality at the end of the time horizon. Theorem 19 shows that it is not the case.

Theorem 19. *A greedy approach does not provide any guarantee of optimality for the T-PFA problem.*

Proof. Consider a fair allocation problem where $\mathcal{X} = \{x \in \mathbb{Z}^3_{\geq 0} : x_1 + x_2 + x_3 = 1\}$. Further consider τ defined as

$$\begin{aligned} [\tau(x)]_1 &= \epsilon x_1 + \frac{1}{2}x_2 + \frac{1}{2}x_3 \\ [\tau(x)]_2 &= x_2 \\ [\tau(x)]_3 &= x_3. \end{aligned}$$

where $0 < \epsilon < \frac{1}{2}$. If $T = 2$, it is obvious that the only perfectly fair solution (and, as it happens, the most efficient too) is achieved by allocating the resource in turn to x_2 and x_3, over T. Yet, for any reasonable measure of unfairness, the greedy choice would be to allocate the resource twice to x_1. Thus, a greedy approach does not necessarily lead to an optimal solution. ■ □

We should note that there is a trade-off between fairness and efficiency: As illustrated in Example 20, enforcing constraints to increase fairness will generally decrease efficiency, and vice-versa.

Example 20. *Consider a fair allocation problem where $\mathcal{X} = \{x \in \mathbb{Z}^3_{\geq 0} : 1 \leq x_1 + x_2 + x_3 \leq 3\}$. Further consider τ defined as*

$$\begin{aligned} [\tau(x)]_1 &= x_1 + \frac{1}{2}x_2 + \frac{1}{2}x_3 \\ [\tau(x)]_2 &= x_2 + \frac{1}{2}x_1 \\ [\tau(x)]_3 &= x_3 + \frac{1}{2}x_1. \end{aligned}$$

By enforcing maximum efficiency, the only feasible solution is $x = (3, 0, 0)$ for $y = (3, \frac{3}{2}, \frac{3}{2})$ and an efficiency of 6, which is not very fair. In contrast, by enforcing maximum fairness, the only feasible solution is $x = (0, 1, 1)$ for $y = (1, 1, 1)$ and an efficiency of 3, which is not very efficient (one resource is even wasted due to the fairness constraints).

We should also note that the value of T may have a noticeable impact on both fairness and efficiency, as illustrated in Example 21.

Example 21. *Consider a fair allocation problem where $\mathcal{X} = \{x \in \mathbb{Z}^4_{\geq 0} : 1 \leq x_1 + x_2 + x_3 + x_4 \leq 3\}$. Further consider τ defined as*

$$\begin{aligned} [\tau(x)]_1 &= x_1 + x_2 \\ [\tau(x)]_2 &= x_2 + x_1 \\ [\tau(x)]_3 &= x_3 + x_4 \\ [\tau(x)]_4 &= x_4 + x_3. \end{aligned}$$

The smallest T accommodating perfect fairness is $T = 1$ with $x = (1, 0, 1, 0)$[3] for $y = (1, 1, 1, 1)$ and an efficiency of 4 (with one wasted resource). However, $T = 2$ also accommodates solutions with perfect fairness,

[3] Or any other equivalent solution.

albeit with an increased efficiency. Take, for instance, $x = (3, 0, 0, 0)$ *and* $x = (0, 0, 0, 3)$ *over two time points, for a combined* $y = (3, 3, 3, 3)$ *and an efficiency of 12. This represents a noticeable increase in efficiency over choosing the initial solution twice to cover the same time horizon.*

In other words, reaching perfect fairness in the shortest possible horizon might not lead to the most efficient solutions.

5 Ambulance Location and Relocation

The task of allocating ambulances to a set of bases in a region is a well-known problem in operations research (Brotcorne et al., 2003). The objective of this problem is generally one of efficiency: Ambulances should be allocated to prime spots such that they can quickly reach the maximum number of people.

The definition of this problem is not unified across the literature (Brotcorne et al., 2003, Gendreau et al., 2001).[4] Most definitions, however, agree that the population should receive an efficient service, which is generally translated into some form of coverage. In this paper, we are not solely concerned by efficiency, but we are further interested in fairness: Ambulances should be allocated such that the same set of people is not always at a disadvantage with respect to access to a quick service.

5.1 Preliminary problem formulation

In this manuscript, we adopt the following definition. The region to which ambulances are allocated is divided into n demand zones, the travel time between each pair being a known quantity. Ambulances can only be allocated to bases, which are located within a subset of the zones. Each zone (equivalently, the people living in each zone) are individual stakeholders in this model. A pre-chosen T rounds of decision is modeled to occur, over which the ambulances should be allocated in a fair and efficient manner. In each round, a configuration x of how m ambulances are placed is chosen. Next, we define the benefit function for the zones. In this model, each zone i, based on its population, has a demand of ζ_i ambulance vehicles. A zone i is said to be *covered* in configuration x if there are at least ζ_i ambulances located in bases from which zone i could be reached in a time less than a chosen time limit called the *response threshold time*. Now, $[\tau(x)]_i = 1$ if zone i is covered by the configuration x and $[\tau(x)]_i = 0$ otherwise. We note that we use a nonlinear benefit function in this case, as opposed to the simpler cases in Section 3 to both reflect the fact that more number of ambulances are essential to sufficiently serve certain regions and also to demonstrate the robustness of our methods to even certain classes of nonlinear (piecewise linear) benefit functions. The nonlinearity, as shown below, can be modeled using auxiliary variables and linear functions, to conform to the form of (8). Furthermore, efficiency constraints analogous to (7c) are enforced by stating that at least a fraction f of the zones should be covered in each allocation. The unfairness metric ϕ used is the difference between the two zones which are most often and least often covered, over T.

With the above, the problem is a standard T-PFA problem in Definition 8. However, the ambulance allocation problem involves additional constraints on allocations between two consecutive periods. Decision-makers prefer policies that ensure that not too many ambulances have to shift bases on a daily basis. Thus, between two consecutive allocations, we would ideally like to have not more than a fixed number r of ambulances to shift bases. We refer to this constraint as the *transition constraint*.

Now, the problem can be formally cast as follows:

$$\min_{y,v,x,\tau,\phi} \quad \phi(y) \tag{9a}$$

$$\text{s.t.} \quad x_i(t) = 0 \qquad \forall\, i \notin \mathcal{B};\, \forall\, t \in \mathscr{T} \tag{9b}$$

$$\sum_{i=1}^{n} x_i(t) \leq m \qquad \forall\, t \in \mathscr{T} \tag{9c}$$

[4] This is due to the fact that research projects often work with datasets associated with specific regions, each of which must follow local guidelines and regulations.

$$v_i(t) = \sum_{j=1}^{n} a_{ji} x_j(t) \qquad \forall\, i = 1, \ldots, n;\ t \in \mathscr{T} \tag{9d}$$

$$v_i(t) \leq (\zeta_i - 1) + m\,\tau_i(t) \qquad \forall\, i = 1, \ldots, n;\ t \in \mathscr{T} \tag{9e}$$

$$v_i(t) \geq \zeta_i - m(1 - \tau_i(t)) \qquad \forall\, i = 1, \ldots, n;\ t \in \mathscr{T} \tag{9f}$$

$$y_i = \frac{1}{T} \sum_{t=1}^{T} \tau_i(t) \qquad \forall\, i = 1, \ldots, n \tag{9g}$$

$$\sum_{i=1}^{n} \tau_i(t) \geq fn \qquad \forall\, t \in \mathscr{T} \tag{9h}$$

$$x_i(t) \in \mathbb{Z}_{\geq 0} \qquad \forall\, i = 1, \ldots, n;\ t \in \mathscr{T} \tag{9i}$$

$$\tau_i(t) \in \{0, 1\} \qquad \forall\, i = 1, \ldots, n;\ t \in \mathscr{T} \tag{9j}$$

$$\sum_{i=1}^{n} |x_i(t+1) - x_i(t)| \leq 2r \forall\, t \in 1, \ldots, T-1 \tag{9k}$$

Here, $\mathscr{T} = \{1, \ldots, T\}$, $x_i(t)$ is the number of ambulances allotted to zone i at time t. Constraints (9b) ensure that allocation occurs only if i is an ambulance base. Here, $\mathcal{B}$ is the set of ambulance bases. Constraints (9c) limit the number of ambulances available for allocation in each round. Moreover, a_{ij} is a binary parameter which is 1 if an ambulance can go from i to j within the response threshold time and 0 otherwise. Through constraints (9d), $v_i(t)$ counts the number of ambulances that can reach zone i within the response threshold time. Here, $\tau_i(t)$ is 1 if $v_i(t)$ is at least ζ_i, i.e., if the demand of ambulances in zone i is satisfied, and 0 otherwise. This is accomplished by constraints (9e) and (9f). Constraints (9h) take care of efficiency and ensure only allocations that cover at least a fraction f of all zones are to be considered. Finally, constraints (9k) are the transition constraints, which can easily be reformulated through linear inequalities. It ensures that not too many ambulances shift bases between consecutive time periods.

The problem in (9) is a mixed-integer linear program (MILP). However, the problem is symmetric with respect to certain permutations of the variables. Symmetry makes it particularly hard for modern branch-and-bound-based solvers (Margot, 2010). Even if one relaxes the transition constraints (9k), the symmetry exists. One can observe that, relaxing the transition constraints (9k) in (9), we have a T-PFA problem. We call this relaxed problem defined by (9a) to (9j) as the Ambulance-without-transition constraints (AWT) problem.

5.2 A branch-and-price reformulation of the AWT

Since the AWT in (9a) to (9j) is a T-PFA problem, it can be readily written in the form shown in (8) and hence can be solved using branch-and-price. First, we note that without the constraint in (9k), any feasible solution or optimal solution remains feasible or optimal after permutations to t. Thus, the following version of the problem could be used to solve the relaxed problem without the symmetry. Here we only *count* the number of times each configuration might be used over T periods.

The master problem (MP).

$$\min \quad g - h \tag{10a}$$

$$\text{s.t.} \quad g \geq y_i \qquad (\alpha_i) \quad \forall\, i = 1, \ldots, n \tag{10b}$$

$$y_i \geq h \qquad (\beta_i) \quad \forall\, i = 1, \ldots, n \tag{10c}$$

$$y_i = \frac{1}{T} \sum_{j=1}^{k} \tau_{ij}\, q_j \qquad (\lambda_i) \quad \forall\, i = 1, \ldots, n \tag{10d}$$

$$\sum_{j=1}^{k} q_j = T \qquad (\mu) \tag{10e}$$

$$q_j \geq 0 \qquad \forall\, j = 1, \ldots, k \tag{10f}$$

$$q_j \in \mathbb{Z} \qquad \forall\, j = 1, \ldots, k. \tag{10g}$$

where q_j counts the number of times the configuration defined by x^j is used. The benefit obtained by stakeholder i due to the configuration x^j is τ_{ij}. y_i is the average benefit that stakeholder i enjoys through the time horizon of planning. Note that, once we solve (10), an equivalent solution to the AWT could be obtained by arbitrarily considering the allocations x^j for q_j times in $x(1), \ldots, x(T)$ of (9). Similarly given a solution to AWT, one could immediately identify a corresponding solution to (10).

However, since the number of configurations are typically exponentially large and we only might use a handful of them in a solution, we could resort to a branch-and-price approach where (10) only contains a subset of the configurations.

Referring to the continuous relaxation of MP as CMP, we write the dual of CMP below.

$$\max \quad T\mu \tag{11a}$$

$$\text{s.t.} \quad \sum_{i=1}^{n} \alpha_i = 1 \qquad (g) \tag{11b}$$

$$\sum_{i=1}^{n} \beta_i = 1 \qquad (h) \tag{11c}$$

$$\lambda_i - \alpha_i + \beta_i = 0 \qquad (y_i) \qquad \forall\, i = 1, \ldots, n \tag{11d}$$

$$\mu \leq \frac{1}{T} \sum_{i=1}^{n} \tau_{ij}\, \lambda_i \qquad (q_j) \qquad \forall\, j = 1, \ldots, k \tag{11e}$$

$$\alpha_i,\ \beta_i \geq 0 \qquad \forall\, i = 1, \ldots, n. \tag{11f}$$

Considering only a subset of columns in the CMP is equivalent to considering only a subset of constraints in (11e). Given some dual optimal solution $(\alpha^\star, \beta^\star, \lambda^\star, \mu^\star)$ to the dual of the restricted CMP, one can find the most violated constraint in (11e) and include the corresponding column in the restricted CMP.

The pricing problem.

$$\min \quad \sum_{i=1}^{n} \tau_i\, \lambda_i^\star \tag{12a}$$

$$\text{s.t.} \quad (\tau_i = 1) \iff (a_i^\mathsf{T} x \geq \zeta_i), \qquad \forall\, i = 1, \ldots, n \tag{12b}$$

$$(\tau_i = 0) \iff (a_i^\mathsf{T} x \leq \zeta_i - 1), \quad \forall\, i = 1, \ldots, n \tag{12c}$$

$$\sum_{i=1}^{n} x_i \leq m \tag{12d}$$

$$\sum_{i=1}^{n} \tau_i \geq fn \tag{12e}$$

$$\tau \in \{0, 1\}^n \tag{12f}$$

$$x \in \mathbb{Z}^n_{\geq 0}. \tag{12g}$$

The minimal efficiency constraints are embedded in the pricing problem. The time horizon is fixed in (10e).

Constraints (12b) and (12c) help compute the benefits to each zone, τ and can clearly be rewritten with integer variables and linear constraints. No more than m ambulances may be used (12d), and the configuration must cover at least a fraction f of the zones (12e).

The reformulation (10) of the AWT in (9a) - (9j), can now be solved very efficiently using branch-and-price. However, a solution thus obtained might violate (9k). Further, given the solution in the space of variables in (10) it is not immediate if one can easily verify whether the constraint (9k) is or is not satisfied. So, given a feasible point for (10), we define the configuration graph as follows, and then show that the point satisfies (9k) if and only if the configuration graph has a Hamiltonian path.

5.3 Checking (9k)

Definition 22 (Configuration graph)**.** *Given a solution $\overline{q}$ to (10), define $\mathcal{I} = \{j : \overline{q}_j \geq 1\}$. The configuration graph (CG) is defined as $G = (V, E)$, where $V = \{(j, j') : j \in \mathcal{I}; j' \in \{1, 2, \ldots, \overline{q}_j\}\}$ and $E = \{((j_1, j_1'), (j_2, j_2')) : \left\|\overline{x}^{j_1} - \overline{x}^{j_2}\right\|_1 \leq 2r\}$ where $\overline{x}^{j_1}$ and $\overline{x}^{j_2}$ are the configurations corresponding to the variables $\overline{q}_{j_1}$ and $\overline{q}_{j_2}$.*

Theorem 23. *Given a point $\overline{q}$ that is feasible to (10), there exists a corresponding $\overline{x}$ that is feasible to (9) if the CG defined by $\overline{q}$ contains a Hamiltonian path. Conversely, if there is a feasible solution to (9) which uses only the configurations with indices in $\mathcal{I} = \{j : \overline{q}_j \geq 1\}$, then the corresponding CG has a Hamiltonian path.*

Proof. Observe that G has exactly T vertices, since if a configuration is found more than once in $\overline{q}$, it is split into as many distinct vertices in V, which are distinguished by the second element of the tuple. An edge $((u_1, u_2), (v_1, v_2))$ in E indicates that movement between configurations indexed by u_1 and v_1 does not violate the transition constraints. A Hamiltonian path is a path that visits each vertex exactly once. As such, any Hamiltonian path in G proves that a feasible sequence of transitions which does not violate the transition constraints exists between the configurations in $\overline{q}$. Given a Hamiltonian path $(v_1, w_1), (v_2, w_2), \ldots, (v_T, w_T)$ in G, a feasible sequence $x(1), x(2), \ldots, x(T)$ for (9) would be $\overline{x}^{v_1}, \overline{x}^{v_2}, \ldots, \overline{x}^{v_T}$.

Conversely, given a feasible sequence $x^{j_1}, x^{j_2}, \ldots, x^{j_T}$ for (9), a Hamiltonian path can be constructed for G as $(j_\alpha, \beta) : \alpha \in \{1, \ldots, T\}, \beta = 1 + \max_{\beta'}((j_{\alpha'}, \beta') : \alpha' \in \{1, \ldots, \alpha - 1\} \wedge x^{j_\alpha} = x^{j_{\alpha'}}))$. ■ □

Following Theorem 23, one can construct the CG with exactly T vertices and can check if the solution to (10) is feasible to (9). If yes, we are done. If not, the way we can proceed is detailed in the rest of this section.

5.3.1 Cutting planes and binarization

The most natural way to eliminate a solution that does not satisfy a constraint is by means of a cutting plane. This is a common practice in the computational MILP research, for example, the subtour elimination inequalities in a travelling salesman problem.

In cases where a more sophisticated cutting plane is not available, but every feasible solution is determined by a binary vector, no-good cuts could be used to eliminate infeasible solutions one by one (D'Ambrosio et al., 2010). However, the variables x in (10) are general integer variables. It is possible that a point x that violates the transition constraint could lie strictly in the convex hull of solutions that satisfy the transition constraint. Hence it could be impossible to separate x using a cutting plane. Example 24 demonstrates the above phenomenon.

Example 24. *Consider the case where (10) and (12) has an optimal solution given by $q = (q_1, q_2, q_3, q_4) = (1, 1, 1, 1)$ and the allocations x_1, x_2, x_3, x_4 (in the context of (9)) corresponding to q_1, q_2, q_3, q_4 are $(3, 3, 3, 3)$, $(4, 2, 3, 3)$, $(2, 4, 3, 3)$, $(3, 3, 2, 4)$ respectively. Let $r = 1$. Then, the CG corresponding to this solution is a tree as shown in Figure 1 and hence does not have a Hamiltonian path. On the other hand, for the choices q^i, q^{ii}, q^{iii} and q^{iv} being $(4, 0, 0, 0)$, $(0, 4, 0, 0)$, $(0, 0, 4, 0)$, $(0, 0, 0, 4)$, the CGs are all K_4, i.e., complete graphs and hence have a Hamiltonian path, trivially.*

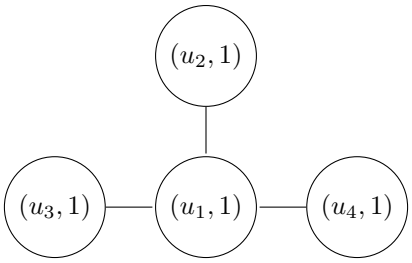

Figure 1: Configuration graph for the solution in Example 24.

Now, it is easy to see that q lies in the convex hull of q^i, q^{ii}, q^{iii} and q^{iv} and while each of the latter solutions satisfies the transition constraint, q does not. Thus no valid cutting plane can cut only the infeasible point.

While Example 24 shows that an infeasible solution cannot always be separated using cutting planes, it could still be possible that there is an extended formulation where the infeasible point could be separated.

Naive binarization. A natural choice of an extended formulation comes from binarizing each integer variable q_j in (10). This is possible because each q_j is bounded above by T and below by 0. Thus one can write constraints of the form $q_j = b_j^1 + 2b_j^2 + 4b_j^4 + 8b_j^8 + \dots$ where the summation extends up to the power of 2 lesser than or equal to T. If each b_j^ℓ is binary, any integer between 0 and T can be represented as above. Having written the above binarization scheme, one can separate any solution q by adding a no-good cut on the corresponding binary variables. A no-good cut is a linear inequality that separates a single vertex of the $0-1$ hypercube without cutting off the rest (Balas and Jeroslow, 1972, D'Ambrosio et al., 2010). An example is shown in Example 25.

Example 25. *Consider the problem in Example 24. Binarization to separate the solution $q = (1, 1, 1, 1)$ can be done by adding the following constraints to (10).*

$$q_j = b_j^1 + 2b_j^2 + 4b_j^4 \qquad \forall\, j = 1, \dots, 4 \tag{13a}$$

$$\sum_{j=1}^{4} \left(b_j^1 + (1 - b_j^2) + (1 - b_j^4)\right) \leq 11 \tag{13b}$$

Note that the second constraint above (the no-good constraint) is violated only *by the binarization corresponding to the solution $u = (1, 1, 1, 1)$ and no other feasible solution is cut off.*

The potential downside with the above scheme is that one might have to cut off a prohibitively large number of solutions before reaching the optimal solution. And with the column generation introducing new q-variables, this could lead to an explosion in the number of new variables as well as the number of new constraints.

Strengthened binarization. When the CG corresponding to q is not just lacking a Hamiltonian path, but is a disconnected graph (a stronger property holds), one could add a stronger cut, which could potentially cut off multiple infeasible solutions. In this procedure, we add a binary variable b_j for each q_j such that $b_j = 1$ if and only if $q_j \geq 1$. Now, observe that if a CG corresponding to the solution $\overline{q}$ is disconnected, then it will be disconnected for all q whose non-zero components coincide with the non-zero components of $\overline{q}$. i.e., no solution with the same support as that of $\overline{q}$ could satisfy the transition constraints. Thus one could add a no-good cut on these b_j binary variables, which cuts off all the solutions with the same support as $\overline{q}$.

Unlike naive binarization, while this cuts off multiple solutions simultaneously, it could happen that the CG is connected, but just does not have a Hamiltonian path. In such a case, no cut could be added by the strengthened binarization scheme, and one might have to resort to the naive version.

5.3.2 Three-way branching

An alternative to the naive binarization scheme is three-way branching. While this could work as a stand-alone method, it could also go hand in hand with the strengthened binarization mentioned earlier. This method takes advantage of the three-way branching feature that some solvers, for example, SCIP (Gamrath et al., 2020a,b), have.

In this method, as soon as a solution $\overline{q}$ satisfying all the integrality constraints but violating the transition constraint is found, the following actions are performed. First, if the lower bound and the upper bound for every component of $\overline{q}$ match, then we are at a leaf that can be discarded as infeasible. If not, find a component j (in our case, a configuration j) such that $\overline{q}_j$ is strictly different from at least one of the bounds. Now, we do a three-way branching on the variable q_j where the new constraints in each of the three branches are (i) $q_j \leq \overline{q}_j - 1$ (ii) $q_j = \overline{q}_j$ (iii) $q_j \geq \overline{q}_j + 1$.

Finite termination follows from the fact that $\overline{q}_j$ is infeasible for branches (i) and (iii) and hence will never be visited again. For branch (ii), we have q_j such that its lower and upper bounds are equal to $\overline{q}_j$. Thus, we have one more fixed variable and this variable will never be branched on again.

Three-way branching is used as opposed to regular two-way branching but on integer variables because of the following reasons. First, branching with (i) $q_j \leq \overline{q}_j - 1$ (ii) $q_j \geq \overline{q}_j + 1$ is invalid, as it could potentially cut off other feasible solutions with $q_j = \overline{q}_j$ but the solution differing in components other than j. Branching with (i) $q_j \leq \overline{q}_j - 1$ (ii) $q_j \geq \overline{q}_j$ could cycle, as we have not fixed any additional variable in the second branch, nor have we eliminated the infeasible solution. Thus, the LP optimum in the second branch is going to be $\overline{q}$ again, and cycling ensues.

5.3.3 Constraint Programming

The final alternative we can use to enforce transition constraints (9k) is constraint programming (CP). Namely, CP is a programming paradigm for solving combinatorial problems. A CP model is defined by a set of variables, each of which is allowed values from a finite set, its *domain*. The relationship between these variables is determined by the constraints of the problem. These succinct constraints can encapsulate complex combinatorial structures, such as the packing of items into bins, for example. A solver then solves the problem by enforcing consistency between the variables and the constraints, and using branching and backtracking techniques to explore the solution space. Before defining the CP model to enforce constraints (9k), we define the compact configuration graph, and explain its relationship with the CG.

Definition 26 (Compact configuration graph)**.** *Given a feasible solution $q^\star$ to the continuous relaxation of (10) (CMP), the compact configuration graph (CCG) is defined as $G = (V, E)$, where $V = \mathscr{A} := \{j : q^\star_j > 0\}$ and $E = \{(v, w) : \|q^\star_v - q^\star_w\|_1 \leq 2r\}$.*

Definition 27 (Walk)**.** *A* walk *in an undirected graph $G = (V, E)$ is a finite sequence of vertices $v_1, \ldots, v_k$, not necessarily distinct, such that for each i, $(v_i, v_{i+1}) \in E$.*

Theorem 28. *Every walk of length T in a CCG constructed from a solution $q^\star$ to CMP corresponds to a feasible solution of (9).*

Proof. Let $G = (V, E)$ be the CCG given a solution $q^\star$. Let $W = v_1, v_2, \ldots, v_T$ be a walk of length T in G. Since we allow revisiting vertices, it is possible that $v_j = v_{j'}$ for some $j \neq j'$.

Now define $\widetilde{q}$ component-wise where $\widetilde{q}_j$ corresponds to the number of times the vertex j is visited in the walk W. Since the walk has a length T, trivially $\sum_j \widetilde{q}_j = T$, satisfying (10e). Then, $\widetilde{y}, \widetilde{q}$ can be defined so that we have a feasible solution to (10). Hence, if we now show that the CG defined by the nonzero components of $\widetilde{q}$ has a Hamiltonian path, then the corresponding $\widetilde{x}$ will be feasible for (9) due to Theorem 23.

Now, in the CG, construct the path $P = (v_1, n(v_1) + 1), (v_2, n(v_2) + 1), \ldots, (v_T, n(v_T) + 1)$ where $n(v_j)$ records the number of times v_j has appeared in the path P earlier so far. We note that each term in the path P is indeed a vertex of the CG (which has T vertices) and that they are all visited exactly once, implying that P is the required Hamiltonian path. ■ □

The general mechanism involving the CP component is provided in Algorithm 1, which is the entire algorithm we test in Section 6. The CP component checks whether the solution returned by the CMP can be made valid in some way, i.e., if there exist solutions using only the configurations in $\mathscr{A}$ (i.e., the configurations that appear in the optimal CMP solution, see Definition 26) with integer values, such that they do not violate the transition constraints. By providing these feasible solutions, it provides an upper bound to (9). As soon as a set of configurations is given to CP, a cut is added to the CMP, which eliminates all solutions to CMP which only consist of the configurations provided to CP. Namely, if $\mathscr{A}$ indices the configurations given to CP, we add a cut $\sum_{j\in\mathscr{A}} q_j \leq T-1$, indicating that at least one of the configurations must be outside the set indexed by $\mathscr{A}$.

Let k' be the cardinality of $\mathscr{A}$. A CCG is associated with solution q — this CCG forms the basis for a deterministic finite automaton. This automaton A is defined by a tuple $(Q, \Sigma, \delta, q_0, F)$ of states Q, alphabet Σ, transition function $\delta : Q \times \Sigma \to Q$, initial state $q_0 \in Q$, and final states $F \subseteq Q$, with $Q = \{0, \dots, k'\}$, $\Sigma = \{1, \dots, k'\}$, $\delta = \{(u, v) \to v : (u, v) \in E\} \cup \{(0, u) \to u : u \in F\}$, $q_0 = 0$, and $F = \{1, \dots, k'\}$. In other words, this automaton accepts any valid sequence of k' configurations, with dummy configuration $q_0 = 0$ being the initial state. Let LB be the lower bound given by the CMP, and UB be an upper bound. Finally, let Ω be the collection of all index sets $\mathscr{A}'$ for which the CP model has been previously solved. There are T decision variables z with domains $\{0, \dots, k'-1\}$, with z_t indicating which CMP configuration from $\mathscr{A}$ is used at time point t. These variables are constrained by

$$\min \phi \tag{14a}$$
$$\phi = \texttt{max}(c) - \texttt{min}(c) \tag{14b}$$
$$LB \leq \phi \leq UB - 1 \tag{14c}$$
$$\texttt{cost\_regular}(z, A, \tau_{i\star}, c_i) \qquad i = 1, \dots, n \tag{14d}$$
$$\texttt{at\_most}(T-1, z, \omega) \qquad \forall \omega \subset \mathscr{A} : \exists \mathscr{A}' \in \Omega,\ \omega \subseteq \mathscr{A}' \tag{14e}$$
$$z_t \in \mathscr{A} \qquad t = 1, \dots, T \tag{14f}$$
$$c_i \in \mathbb{Z}_{\geq 0} \qquad i = 1, \dots, n \tag{14g}$$

The objective (14a) is to minimize the unfairness, i.e., the difference between the zone which is covered the most, and that which is covered the least (14b). Any feasible solution should be strictly better than the upper bound, and search by the CP solver can be interrupted as soon as a feasible solution is found whose objective value coincides with LB (14c). By Theorem 28, any sequence of configurations indexed by $\mathscr{A}$ and of length T must correspond to a feasible solution of (9). This requirement is enforced by the `cost-regular` (Demassey et al., 2006) constraints (14d): A `cost-regular` constraint holds for zone i if z forms a word recognized by automaton A and if variable c_i is equal to the coverage of zone i over T: $c_i = \sum_{t=1}^{T} \tau_{i,z_t}$. Note that the CP model does not require that all configurations indexed by $\mathscr{A}$ be used at least once: Since it checks all possible subsets of $\mathscr{A}$, the cut added to CMP does not remove any unexplored part of the search space. Finally, all subsets $\omega \subset \mathscr{A}$ previously explored, i.e. being as well a subset of some index set $\mathscr{A}'$ in Ω, are considered by the model using the `at_most` constraint (14e): At most $T-1$ variables in z can take on values in ω. The motivation for this is illustrated in Example 29.

Example 29. *Assume that $\mathscr{A} = \{j', j'', j'''\}$. If (14) had previously solved $\mathscr{A}' = \{j', j''\}$, this solution space would be explored again since the configurations in $\mathscr{A}$ are not constrained to be used at least once. Adding constraint $\texttt{at\_most}(T-1, z, \{j', j''\})$ in the current iteration avoids this.*

The CP solver we use is OR-Tools, which currently does not implement the `cost-regular` constraint. We thus replaced (14b) and (14d) with the equivalent but less efficient[5] reformulation

$$\phi = \max_{i=1}^{n} \sum_{t=1}^{T} \tau_{i,z_t} - \min_{i=1}^{n} \sum_{t=1}^{T} \tau_{i,z_t}$$

[5]In practice, the CP component of Algorithm 1 remains very fast, so this loss in efficiency is negligible.

Algorithm 1 The Final algorithm

Input: The ambulance allocation problem with number of zones n, the bases $\mathcal{B}$, ζ_i, for $i = 1, \ldots, n$, f and a_{ih} for $i, h = 1, \ldots, n$, $r \in \mathbb{Z}_{\geq 0}$ and $T \in \mathbb{Z}_+$.

Output: $x(1), \ldots, x(T)$ that is optimal to (9).

1: $LB \leftarrow -\infty$, $UB \leftarrow +\infty$. $\mathscr{C} \leftarrow \emptyset$. $x^\star(1), \ldots, x^\star(T) = NULL$.
2: **while** $UB > LB$ **do**
3: Solve CMP, i.e., the continuous relaxation of (10) after adding each constraint in $\mathscr{C}$. Let $q^\star$ be the optimal solution and $o^\star$ be the optimal objective value.
4: $LB \leftarrow o^\star$.
5: $\mathscr{A} \leftarrow \{j : q^\star_j > 0\}$.
6: $\mathscr{C} \leftarrow \mathscr{C} \cup \left\{ \sum_{j \in \mathscr{A}} q_j \leq T - 1. \right\}$.
7: $(x^\dagger(1), \ldots, x^\dagger(T)), o^\dagger \leftarrow$ ConstraintProgramming$(q^\star, n, a, T, LB, UB, \mathscr{C})$.
8: **if** $o^\dagger < UB$ **then**
9: $UB \leftarrow o^\dagger$ and $(x^\star(1), \ldots, x^\star(T)) \leftarrow (x^\dagger(1), \ldots, x^\dagger(T))$.
10: **end if**
11: **end while**
12: **return** $x^\star(1), \ldots, x^\star(T)$
13:
14: **function** ConstraintProgramming$(q, n, a, T, LB, UB, \mathscr{C})$
15: $G \leftarrow$ CCG defined by $\mathscr{A}$.
16: Solve (14) on the graph G with appropriate values of τ, c.
17: **if** (14) is infeasible **then**
18: **return** $NULL$, $+\infty$
19: **else**
20: **return** $(x^\dagger(1), \ldots, x^\dagger(T)), o^\dagger$.
21: **end if**
22: **end function**

$$\texttt{regular}(z, A).$$

5.4 The Final Algorithm

The general working of the final algorithm is presented formally in Algorithm 1.

In each iteration of the final algorithm, we solve a linear program and make a call to the constraint programming solver. The linear program is basically the continuous relaxation of (10) with any subsequent cutting planes (Step 6 in Algorithm 1). This is solved using column generation. The configurations which receive non-zero weights (indexed by $\mathscr{A}$) in the optimal LP solution are passed to the constraint programming solver to detect whether there exists a solution to (9) that uses only configurations indexed by $\mathscr{A}$. Meanwhile, a cut is added to the linear program, that eliminates all solutions which use only the configurations indexed by $\mathscr{A}$.

With this, the large set of possible configurations are managed by the linear programming solver, which is powerful and efficient for large problems, while the CP-based solver only solves instances with a handful of configurations at a time.

6 Computational experiments

In this section, we introduce the setting that we have used to evaluate computationally Algorithm 1 and we discuss the results of such an evaluation in the context of ambulance location and relocation. In particular,

we use real data from the city of Utrecht, Netherlands, as well as synthetically-generated instances of varying sizes. Moreover, we consider a predetermined time horizon, i.e., a fixed T of size 30, with the understanding that a decision maker could reasonably make plans on a monthly basis.

Utrecht instance. We use the model defined in Section 5 to determine ambulance allocation in the city of Utrecht, Netherlands, using a dataset provided by the RIVM.[6] The Utrecht instance contains 217 zones, 18 of which are bases. Since calls should be reached within 15 minutes, we consider that a zone is connected to another if an ambulance can travel between the two zones in under 15 minutes. This makes the graph about 28.4% dense.

The efficiency measure we impose is that at least 95% of all the zones should be covered at all times. We consider that a zone is covered if sufficiently many ambulances are in the vicinity of that zone. In turn, we define the sufficient number of ambulances for a zone to be 1, 2, 3, or 4, based on the population density of the zone.

Synthetic instances. Reproducing the ratio of bases and edges to zones, we also generated synthetic instances[7] of sizes 50, 100, 200, and 400 zones, using the Matérn cluster point process (Matérn, 1960), which helps in generating a distribution of zones mimicking a realistic urban setting. The number of ambulances for the instances is chosen to be just enough to ensure feasibility. The results are averaged over five instances of each size.

Testing environment and software. Testing was performed on an Intel 2.8 GHz processor with 8 GBs of RAM running Arch Linux, and the models were solved with Gurobi 9.0.1 and OR-Tools 8.0. A time limit of 1,200 seconds was imposed.

Results. It is easy to see that there are two parameters that are likely to influence the difficulty of the solving process: the size of the instance (number of zones) and the transition constraints, i.e., the flexibility we allow for relocating ambulances between zones on a daily basis. We would typically expect the size of the instance to be a significant factor, but this is likely to be mitigated by the column generation approach, which tends to scale well. The effects that the transition constraints will have on the solving process, however, are less clear. In order to assess such an effect, for each instance, we vary the number of ambulances that can shift bases on consecutive days (r in constraints (9k)) from $0.1m$ to m in increments of $0.1m$, where m is the total number of ambulances in the instance. We call this ratio maximum transition, MT. For each of these cases, we record the average of the time taken to solve these instances to optimality (capped at 1,200 CPU seconds). We record the final relative gap left for the instance, which is defined by $\frac{|UB_f - LB_f|}{UB_f}$, where LB_f and UB_f represent the values of the best lower and upper bounds at the time limit, respectively. We also record the initial relative gap for the instance, which is defined by $\frac{|UB_i - LB_i|}{UB_i}$, where LB_i represents the value of the initial lower bound (without any cuts), and UB_i represents the value of the initial, trivial upper bound. We present the final relative gaps associated with different classes of instances in Figure 2.

One can immediately observe from Figure 2 that the transition constraints are affecting the difficulty of the instances for $MT \leq 0.5$, while everything can be solved to optimality within the time limit for $MT \geq 0.5$ independently of the number of zones. The larger MT the smaller the final gap, i.e., when there is more freedom in moving the ambulances around on a daily basis, Algorithm 1 becomes very effective in computing optimal solutions.

We observe that for varying values of MT, the Utrecht instance has a higher relative gap than synthetic instances of comparable and even larger size. This discrepancy is the result of the distribution of the population densities across the zones. In the synthetic instances, the population densities are randomly

[6] National Institute for Public Health and the Environment of the Netherlands.

[7] These instances can be found at https://github.com/PhilippeOlivier/ambulances.

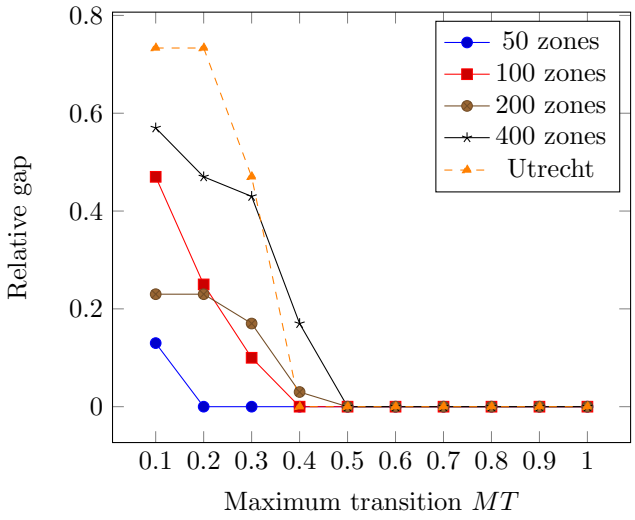

Figure 2: Final relative gaps with respect to the maximum transition MT. Gaps in the synthetic instances are averaged over 5 instances. Time limit of 1,200 CPU seconds.

distributed among the zones. The Utrecht instance, in contrast, exhibits several distinct clusters of varying sizes and population density distributions.[8]

Unsurprisingly, Figure 2 also suggests that larger instances are more difficult because, typically, the relative gap is large when time limits are hit. This is confirmed in more details from the results in Table 1, where we report, for every group of instances and every MT value, the initial gap ("i.gap"), the final gap at the time limit ("f.gap"), the number of generated columns ("#cols"), the number of solved instances ("#solved"),[9] and the average time for the instances solved to optimality ("time").

	50 zones					**100 zones**					**200 zones**				
MT	i.gap	f.gap	#cols	#solved	time	i.gap	f.gap	#cols	#solved	time	i.gap	f.gap	#cols	#solved	time
0.1	0.13	0.13	738	4	0.3	0.55	0.47	1120	1	0.2	0.23	0.23	714	3	2.5
0.2	0.13	0.00	3	5	0.4	0.55	0.25	674	2	1.2	0.23	0.23	710	3	2.5
0.3	0.13	0.00	3	5	0.4	0.55	0.10	11	4	2.5	0.23	0.17	537	3	2.6
0.4	0.13	0.00	3	5	0.4	0.55	0.00	9	5	2.7	0.23	0.02	108	4	82.7
0.5	0.13	0.00	3	5	0.4	0.55	0.00	9	5	2.3	0.23	0.00	19	5	7.0
0.6	0.13	0.00	3	5	0.4	0.55	0.00	9	5	2.1	0.23	0.00	15	5	7.2
0.7	0.13	0.00	3	5	0.4	0.55	0.00	9	5	2.1	0.23	0.00	15	5	6.7
0.8	0.13	0.00	3	5	0.4	0.55	0.00	9	5	2.2	0.23	0.00	15	5	6.0
0.9	0.13	0.00	3	5	0.4	0.55	0.00	9	5	2.2	0.23	0.00	15	5	6.0
1	0.13	0.00	3	5	0.4	0.55	0.00	9	5	2.1	0.23	0.00	15	5	6.0

	400 zones					**Utrecht**				
MT	i.gap	f.gap	#cols	#solved	time	i.gap	f.gap	#cols	#solved	time
0.1	0.57	0.57	248	1	2.7	0.73	0.73	786	0	-
0.2	0.57	0.47	246	1	2.7	0.73	0.73	803	0	-
0.3	0.57	0.43	238	1	2.7	0.73	0.47	757	0	-
0.4	0.57	0.17	174	3	147.4	0.73	0.00	353	1	315.6
0.5	0.57	0.00	93	5	273.6	0.73	0.00	252	1	122.3
0.6	0.57	0.00	65	5	63.6	0.73	0.00	96	1	49.6
0.7	0.57	0.00	60	5	49.6	0.73	0.00	96	1	50.3
0.8	0.57	0.00	60	5	47.1	0.73	0.00	96	1	48.4
0.9	0.57	0.00	60	5	47.4	0.73	0.00	96	1	49.8
1	0.57	0.00	60	5	49.1	0.73	0.00	96	1	54.7

Table 1: Detailed results for Algorithm 1 on both synthetic and real-world Utrecht instances. Time limit of 1,200 CPU seconds. The results of the synthetic instances are averaged over 5 instances.

The results in Table 1 show that for $MT > 0.7$ constraints (9k) are not binding, i.e., they do not cut off the obtained optimal solution to the rest of the model, and Algorithm 1 behaves exactly like for $MT = 0.7$.

[8] Constructing these sophisticated clusters is not a trivial task, which is why we settled on a random distribution of population densities for the synthetic instances. By randomizing the placement of the population for the Utrecht instance, its gap becomes similar to that of the synthetic instances.

[9] The entry "#solved" takes an integer value between 0 and 5 for synthetic instances and 0/1 for the Utrecht instance.

The average time for the instances solved to optimality is often low, thus indicating that if we manage to prove optimality by completely closing the gap, then this is achieved quickly, generally with a few calls to ConstraintProgramming. In contrast, if we do not succeed in closing the gap early, then it is unlikely to be closed at all in the end. In the cases where optimality is not proven, improvements are still made during the solving process mostly because of improvements on the upper bound value as ConstraintProgramming tests more and more combinations of configurations. This suggests a few things. First, with a high enough MT, the initial lower bound is generally optimal, and feasible solutions whose objective values coincide with this bound can readily be found. Second, when the MT value is low, either feasible solutions whose objective values coincide with the initial lower bound are very rare, or they are nonexistent. Third, the computational power of Algorithm 1 is associated with the effectiveness in searching for primal solutions (upper bound improvement) versus the progress on the dual side (lower bound improvement), which appears to be very difficult.

Finally, we are also interested in identifying the time spent in the column generation part as opposed to the ConstraintProgramming routine. To this end, we track the number of calls made to the function ConstraintProgramming and also measure the total time spent in calls to the function. The number of generated columns and the number of calls to ConstraintProgramming are detailed in Table 2.

	50 zones		**100 zones**		**200 zones**		**400 zones**		**Utrecht**	
MT	#cols	#calls	#cols	#calls	#cols	#calls	#cols	#calls	#cols	#calls
0.1	738	736	1120	1108	714	608	248	72	786	210
0.2	3	1	674	662	710	605	246	72	803	217
0.3	3	1	11	3	537	435	238	66	757	202
0.4	3	1	9	1	108	58	174	22	353	54
0.5	3	1	9	1	19	3	93	6	252	21
0.6	3	1	9	1	15	1	65	2	96	1
0.7	3	1	9	1	15	1	60	1	96	1

Table 2: Number of columns and calls to ConstraintProgramming associated with the maximum transition (MT) constraints of the various instance sizes.

The results in Table 2 show that when enough freedom in the day-to-day movement of ambulances is allowed, optimality can generally be proven immediately, with just one call to the ConstraintProgramming routine. We also notice that the ratio of columns to the number of calls to the routine increases with the instance size. Such a ratio for the Utrecht instance is disproportionately high, due again to the distribution of the population densities.

Finally, Table 3 shows the average amount of time spent generating columns ("CG"), that spent enforcing the transition constraints through ConstraintProgramming ("CP") and the ratio between these two CPU times ("ratio") for the most difficult case, i.e., $MT = 0.1$.

	time		
Instance	CG	CP	ratio
50 zones	86	154	0.56
100 zones	411	550	0.75
200 zones	331	151	2.19
400 zones	711	253	2.81
Utrecht	1071	129	7.96

Table 3: CPU times in seconds spent in column generation and ConstraintProgramming, with $MT = 0.1$.

The results in Table 3 show that generating columns for larger instances (more zones) takes naturally more time, which is why this effort increases with n. The enforcement of the transition constraints, however,

requires a more stable effort, as the input of ConstraintProgramming is always around three to ten variables, regardless of the size of the instance.

7 Conclusion

We introduced an abstract framework for solving a sequence of fair allocation problems, such that fairness is achieved over time. For some relevant special cases, we have been able to give theoretical proofs for the time horizon required for perfect fairness. We described a general integer programming formulation for this problem, as well as a formulation based on column generation and constraint programming. This latter formulation can be used in a practical context, as shown by the ambulance location problem applied to the city of Utrecht. The largest synthetic instances suggest that this formulation would scale well to regions twice the size of Utrecht, given that the freedom of movement of the ambulances is not overly restricted.

Acknowledgements

The authors would like to thank the National Institute for Public Health and the Environment of the Netherlands (RIVM) for access to the data of their ambulance service.

References

Egon Balas and Robert Jeroslow. Canonical cuts on the unit hypercube. *SIAM Journal on Applied Mathematics*, 23(1):61–69, 1972.

Evripidis Bampis, Bruno Escoffier, and Sasa Mladenovic. Fair resource allocation over time. In *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems*, AAMAS '18, page 766–773, Richland, SC, 2018. International Foundation for Autonomous Agents and Multiagent Systems.

Luce Brotcorne, Gilbert Laporte, and Frédéric Semet. Ambulance location and relocation models. *European Journal of Operational Research*, 147(3):451–463, 2003. ISSN 0377-2217. doi: https://doi.org/10.1016/S0377-2217(02)00364-8. URL http://www.sciencedirect.com/science/article/pii/S0377221702003648.

Oscar R. Burt and Curtis C. Harris. Letter to the editor—apportionment of the u.s. house of representatives: A minimum range, integer solution, allocation problem. *Operations Research*, 11(4):648–652, 1963. doi: 10.1287/opre.11.4.648. URL https://doi.org/10.1287/opre.11.4.648.

Sophie Demassey, Gilles Pesant, and Louis-Martin Rousseau. A cost-regular based hybrid column generation approach. *Constraints*, 11(4):315–333, Dec 2006. ISSN 1572-9354. doi: 10.1007/s10601-006-9003-7. URL https://doi.org/10.1007/s10601-006-9003-7.

Claudia D'Ambrosio, Antonio Frangioni, Leo Liberti, and Andrea Lodi. On interval-subgradient and no-good cuts. *Operations Research Letters*, 38(5):341–345, 2010.

Ohad Eisenhandler and Michal Tzur. The humanitarian pickup and distribution problem. *Operations Research*, 67(1):10–32, 2019. doi: 10.1287/opre.2018.1751. URL https://doi.org/10.1287/opre.2018.1751.

Gerald Gamrath, Daniel Anderson, Ksenia Bestuzheva, Wei-Kun Chen, Leon Eifler, Maxime Gasse, Patrick Gemander, Ambros Gleixner, Leona Gottwald, Katrin Halbig, Gregor Hendel, Christopher Hojny, Thorsten Koch, Pierre Le Bodic, Stephen J. Maher, Frederic Matter, Matthias Miltenberger, Erik Mühmer, Benjamin Müller, Marc E. Pfetsch, Franziska Schlösser, Felipe Serrano, Yuji Shinano, Christine Tawfik, Stefan Vigerske, Fabian Wegscheider, Dieter Weninger, and Jakob Witzig. The SCIP Optimization

Suite 7.0. ZIB-Report 20-10, Zuse Institute Berlin, March 2020a. URL http://nbn-resolving.de/urn:nbn:de:0297-zib-78023.

Gerald Gamrath, Daniel Anderson, Ksenia Bestuzheva, Wei-Kun Chen, Leon Eifler, Maxime Gasse, Patrick Gemander, Ambros Gleixner, Leona Gottwald, Katrin Halbig, Gregor Hendel, Christopher Hojny, Thorsten Koch, Pierre Le Bodic, Stephen J. Maher, Frederic Matter, Matthias Miltenberger, Erik Mühmer, Benjamin Müller, Marc E. Pfetsch, Franziska Schlösser, Felipe Serrano, Yuji Shinano, Christine Tawfik, Stefan Vigerske, Fabian Wegscheider, Dieter Weninger, and Jakob Witzig. The SCIP Optimization Suite 7.0. Technical report, Optimization Online, March 2020b. URL http://www.optimization-online.org/DB_HTML/2020/03/7705.html.

Michel Gendreau, Gilbert Laporte, and Frédéric Semet. A dynamic model and parallel tabu search heuristic for real-time ambulance relocation. *Parallel Computing*, 27(12):1641–1653, 2001. ISSN 0167-8191. doi: https://doi.org/10.1016/S0167-8191(01)00103-X. URL http://www.sciencedirect.com/science/article/pii/S016781910100103X. Applications of parallel computing in transportation.

Jessica L. Heier Stamm, Nicoleta Serban, Julie Swann, and Pascale Wortley. Quantifying and explaining accessibility with application to the 2009 h1n1 vaccination campaign. *Health Care Management Science*, 20(1):76–93, Mar 2017. ISSN 1572-9389. doi: 10.1007/s10729-015-9338-y. URL https://doi.org/10.1007/s10729-015-9338-y.

Stephen Jacobsen. On marginal allocation in single constraint min-max problems. *Management Science*, 17 (11):780–783, 1971. doi: 10.1287/mnsc.17.11.780. URL https://doi.org/10.1287/mnsc.17.11.780.

Naoki Katoh, Toshihide Ibaraki, and H. Mine. An algorithm for the equipollent resource allocation problem. *Mathematics of Operations Research*, 10(1):44–53, 1985. URL https://EconPapers.repec.org/RePEc:inm:ormoor:v:10:y:1985:i:1:p:44-53.

Bernard O. Koopman. The optimum distribution of effort. *Journal of the Operations Research Society of America*, 1(2):52–63, 1953. doi: 10.1287/opre.1.2.52. URL https://doi.org/10.1287/opre.1.2.52.

Hanan Luss. *Equitable Resource Allocation: Models, Algorithms and Applications*. Information and Communication Technology Series,. Wiley, 2012. ISBN 9781118449219. URL https://books.google.ca/books?id=Z2_3oWjnASkC.

François Margot. Symmetry in integer linear programming. In *50 Years of Integer Programming 1958-2008*, pages 647–686. Springer, 2010.

Bertil Matérn. Spatial variation – stochastic models and their applications to some problems in forest survey sampling investigations. *Report of the Forest Research Institute of Sweden*, 49(5):1–144, 1960.

Wlodzimierz Ogryczak, Hanan Luss, Michał Pióro, Dritan Nace, and Artur Tomaszewski. Fair optimization and networks: A survey. *Journal of Applied Mathematics*, 2014:612018, Sep 2014. ISSN 1110-757X. doi: 10.1155/2014/612018. URL https://doi.org/10.1155/2014/612018.

Ogryczak, Wlodzimierz. Fair optimization – methodological foundations of fairness in network resource allocation. In *2014 IEEE 38th International Computer Software and Applications Conference Workshops*, pages 43–48, 2014.

Evan L. Porteus and Jonathan S. Yormark. More on min-max allocation. *Management Science*, 18(9): 502–507, 1972. doi: 10.1287/mnsc.18.9.502. URL https://doi.org/10.1287/mnsc.18.9.502.

Naoki Katoh Toshihide Ibaraki. *Resource Allocation Problems: Algorithmic Approaches*, page 1. Foundations of Computing. The MIT Press, 1988. ISBN 0262090279,9780262090278.

Zeev Zeitlin. Minimization of maximum absolute deviation in integers. *Discrete Applied Mathematics*, 3 (3):203 – 220, 1981. ISSN 0166-218X. doi: https://doi.org/10.1016/0166-218X(81)90017-2. URL http://www.sciencedirect.com/science/article/pii/0166218X81900172.