The Computational Complexity of the Provision-after-Wait Problem in Healthcare

A Thesis presented

by

Gowtham Srinivasan

to

The Graduate School

in Partial Fulfillment of the

Requirements

for the Degree of

Master of Science

in

Dept of Computer Science

Stony Brook University

May 2016

Abstract

The Computational Complexity of the Provision-after-Wait Problem in Healthcare

by

Gowtham Srinivasan

Master of Science

in

Computer Science

Stony Brook University

2016

In Provision-after-Wait in Healthcare (PaW), a social planner operating on a constrained budget is required to sponsor medical treatments to a population of patients. Specifically, each patient is allocated to any of the k hospitals to receive a single unit of medical treatment. The cost of treatment depends upon the hospital and is paid for by the planner. Associated with every patient is a set of k non-negative numbers which denote the patients preference or intrinsic value towards getting treated in each of the k hospitals. Since the patients do not pay for the treatment and the planner cannot afford to send each patient to their most preferred hospital, waiting times are used to ration access to over-demanded hospitals. The social planner is responsible for assigning patients and computing the waiting time of each hospital, so that the social welfare is maximized under budget constraints.

It has already been proved that finding optimal equilibrium assignments that maximize social welfare is NP-hard. Besides that, little is known about the complexity of PaW. For instance, it is not clear whether it permits an

FPTAS or is fixed parameter tractable with respect to the number of hospitals. In this work, we study the complexity of PaW and prove it to be NP-hard even if all the values are polynomially bounded in the length of the input. Since PaW is a number problem, this proves that it is NP-hard in the strong sense and, as a consequence, there is no FPTAS for PaW unless P = NP. In addition, we explore various special cases of PaW, their complexity and related algorithms to resolve the problem.

Dedication Page

I would like to dedicate this work and my Masters to my parents L.Srinivasan and S.Jayashree for their support in everything.

Table of Contents

Contents

List of Abbreviations

PaW	Provision after Wait in Healthcare
FPTAS	Fully Polynomial Time Approximation Scheme

Acknowledgements

I would like to thank Professor Jing Chen for her guidance and kindness in this work and many others.

1 Introduction

Healthcare models in the United States, Europe and other places in the world though widely different in many aspects, have a common underlying theme to them. In many cases, there is a social planner(the Govt, insurance company, healthcare service provider etc) who operates under a restricted budget to give medical services to a population of patients. Since the patients do not pay for the medical service, waiting times[4] are used to ration demand. There has been a lot of literature from medical and healthcare experts that aim to reduce waiting times of patients and offer easier access to healthcare. Even though it might not be possible to make everybody happy, is it possible to maximise the happinees of a population by providing them free healthcare under a constrained budget? What can be said about this scenario, computationally?

Provision-after-Wait in Health care[2] gives a computational model for the above described scenario. In the Provision-after-Wait model, we have a population of patients each demanding access to a single unit of medical treatment which is provided at any of the m hospitals. Each patient has a set of preferences about where they want to be treated. Since the patients do not pay for the treatment, their preferences are not influenced by costs and are subjective. Since the access to hospitals are rationed using waiting times[1], the patients know the waiting time of each hospital. Similarly, the social planner knows the set of preference of every patient towards the hospitals.

2 Model

2.1 Description

In this section, we give a formal description of the Provision-after-Wait in healthcare model. This model is the same as in [2] and is described here for convenience. In the following sections, $1 \le i \le n$, $1 \le j \le m$ and $i, j \in \mathbb{Z}^+$

- A population of n patients. Each patient is denoted by P_i, where $1 \le i \le n$ and $i \in \mathbb{Z}^+$
- A set of m hospitals. Each hospital denoted by H_j, where $1 \le j \le m$ and $j \in \mathbb{Z}^+$

- Each hospital has a cost $c_j \in \mathbb{Z}^+$ associated with it.
- Each patient P_i has a value $v_{ij} \in \mathbb{Z}^+$ associated with every hospital j which denotes the patients' preference towards hospital j or the patients' utility for hospital j when treated there immediately.
- A budget B, $B \in \mathbb{Z}^+$ which is the total amount the social planner can spend to provide treatment for the patient population.
- A solution A to the PaW is a pair $A : (a, w)$, where
  - a is a feasible assignment function $a : [n] \to [m]$ which allocates each patient to a particular hospital. An assignment function a is feasible if the total cost of the assignment is less than the budget i.e, $\sum c_{a(i)} \leq B$
  - $w = (w_1, w_2, ., w_m), w_j \geq 0$, is the waiting time vector
  - Under an assignment A, each patient has an *utility* $u_i = v_{ia(i)} - w_{a(i)}$. The social welfare under assignment A, $SW(A) = \sum u_i$

2.2 Equilibrium assignments

An assignment A is stable under the following two conditions

- Each patient has a non-negative utility under the assignment. i.e $u_i = v_{ia(i)} - w_{a(i)}$
- The hospital allocated for each patient under the assignment A should have the maximum utility for the patient compared to all other hospitals. i.e for any hospital j and for each patient i, $v_{ia(i)} - w_{a(i)} \geq v_{ij} - w_j$

2.3 Optimal equilibrium assignments

An assignment $A^{'}$ is an optimal equilibrium assignment if $SW(A^{'}) \geq SW(A)$, where A is any equilibrium assignment.

3 Related Work

[2] introduces the problem of $Provision - after - Wait\ in\ Health\ care$. The authors give a polynomial time reduction from *knapsack* to prove that PaW is NP-hard. In addition, if there is a ϵ deficit in the budget i.e with a budget $B(1+\epsilon)$, the authors give an algorithm with running time $\mathcal{O}((log_{1+\epsilon}m)^k(1+\epsilon)^3m^4)$ which gives an equilibrium assignment with social welfare atleast $SW(A')$, where A' is the optimal equilibrium solution with a budget B. Randomised assignments are discussed and the authors prove that randomised assignments are also optimal with respect to social welfare in many cases. Endogenous emergence of waiting times and other properties of PaW are found.

[3] introduces a slightly different version of PaW called $PaW\ with\ common\ preferences$, where each patient P_i has a value v_i for getting the medical treatment and each hospital has a quality q_j, which is publicly known to all the patients. The value of patient i towards hospital j is v_iq_j. In this paper, the authors show various properties of an assignment function which emerge from the above structure and how they influence social welfare. The authors give a reduction from *subsetsum* to prove that $PaW\ with\ common\ preferences$ is NP-hard. In addition, an FPTAS with a running time of $\mathcal{O}((m+n).n^3m/\epsilon)$ is provided. The concept of patient ordering is introduced and the results of randomised assignments in [2] are extended.

In [7], the authors prove that the $unit\ demand\ envy\ free\ pricing$ problem is hard to approximate by a reduction from $vertex - cover$

4 Strong NP-harndess

When dealing with NP-hard problems, a natural question that arises is 'Are all NP-hard problems equally NP-hard? Are there distinctive features that can classify NP-hard problems into different useful categories?'

4.1 Pseudo Polynomial time algorithms

A good starting place to look for such distinctions might be the 'Partition' problem, which has been widely termed as the 'easiest hard problem'[8].

Given a finite multi set $A = \{a_1, a_2, ...a_n\}$ of n integers such that $a_i \in \mathbb{Z}^+, \forall a_i \in A$, we define the Partition problem as follows,

$$Partition \;=\; \{(A) : \text{ there exists a } A' \subseteq A \text{ such that } \sum_{a_i \in A'} a_i = \sum_{a_i \in A - A'} a_i \;\}.$$

[6], [5] discusses a recurrence relation and an algorithm for solving the partition problem. The running time of the algorithm is $\mathcal{O}(nS)$, where S is the sum of all elements in A. On the surface, this running time might look polynomial but it is not. Using a binary encoding scheme, the number of bits required to represent the *Partition* problem is $\mathcal{O}(n \log S)$. In the worst case, increasing the input by one bit increases the running time by a factor of 2, which is exponential. It is clear that, the running time is not polynomial in the number of bits in the input(input length) but it is polynomial with respect to the numeric value of the input. Algorithms with such running times are called psuedo polynomial time algorithms.

4.2 Definition-Strong NP-hardness

In the above case, if the numeric values in the input are polynomially bounded by the length of the input the pseudo polynomial time algorithm works as good as a polynomial time algorithm. So, *Partition* is NP-hard only when the numeric values are very large(exponentially large) with respect to the length of the input. This distinctive feature can be used to categorise NP-hard problems into two categories. If an NP-hard problem is NP-hard even on instances when the numeric values are polynomially bounded by the length of the input, they are called *Strongly* NP-hard and NP-hard problems which can be solved in polynomial time when the numeric values are bounded by a polynomial function of the length of the input and are NP-hard only on instances when the numeric values are very large when compared to the input length are called *Weakly* NP-hard problems.

In addition to the above definition, there are other interpretations of strong NP-hardness. A problem $\prod$ can be defined to be strongly NP-hard if it is NP-hard even when the inputs are represented in unary.[5] [6]

4.3 Proof techniques for strong NP-hardness

Let us say, we want to prove the strong NP-hardness of a problem $\prod$. Is it enough to give a polynomial time reduction from another NP-hard problem to $\prod$? Normal polynomial time reductions do not prove strong NP-hardness, because they do not take into account or give any information about the numeric values in the resulting instance which is key to the whole classification of strong and weak NP-hardness. Following, are two methods to prove the strong NP- hardness of any problem.

- Let $\prod$ be any problem and $\prod_R$ be the instance of $\prod$ where all the numeric values are polynomially bound by the length of the input. Giving a polynomial time reduction from any NP-hard problem to $\prod_R$ proves that $\prod$ is *Strongly* NP-hard. This type of proof follows by definition.

- Let $\prod$ be any strongly NP-hard problem. A polynomial time reduction from $\prod$ to another problem $\prod^{'}$, where the length and numeric values in the resulting instances of $\prod^{'}$ are polynomially bound by the length of $\prod$ and the maximum numeric value of $\prod$ proves the strong NP hardness of $\prod^{'}$.

A more rigorous treatment of strong NP-hardness can be found on [6] and [5].

4.4 Practical aspects

Now that we have classified NP-hard problems into two distinct categories, we have another question at hand. 'Is this classification purely theoretical? Are there practical applications for this?'

For example, some NP-complete scheduling problems might be intractable only for instances where the numeric values(length of the tasks)[6] are exponentially large and in those cases scheduling them might be impractical. For instances where the length of the tasks might be polynomial functions of the input length, the pseudo polynomial algorithm acts as a polynomial time algorithm for all practical purposes. Thus looking for a pseudo polynomial time algorithm or proving strong NP hardness is an useful endeavour in its own right.

4.5 Properties

Strongly NP-hard problems cannot have an FPTAS unless P=NP[5] [6].

5 The complexity of the Provision-after-Wait Problem in general

By reducing the Vertex Cover problem to the decision version of the Provision-after-Wait problem, we have the following theorem.

Theorem 5.1. *It is strongly NP-hard to compute an optimal stable assignment for the Provision-after-Wait problem in general.*

Proof. Consider the decision version of the general Provision-after-Wait problem:

$$\begin{aligned} DPaW &= \{(c=(c_j)_{j\in[m]}, v=(v_{ij})_{i\in[n],j\in[m]}, B, T) : \text{ there exists a stable} \\ &\qquad \text{budget-feasible assignment } A \text{ such that } SW(A)\geq T\}. \end{aligned}$$

The Vertex Cover problem, which is strongly NP-hard, is defined as follows:

$$\begin{aligned} VC &= \{(G=(V,E),k) : \text{there exists } V'\subseteq V \text{ with } |V'|=k \\ &\qquad \text{such that, for each edge } (u,v)\in E, \{u,v\}\cap V'\neq\emptyset\}, \end{aligned}$$

where G is an undirected simple graph with vertex set V and edge set E. Given an instance (G,k) of VC with t vertices and e edges, and letting $V=\{1,\ldots,t\}$ and $E=\{(u_1,v_1),...,(u_e,v_e)\}$, we construct an instance (c,v,B,T) of $DPaW$ as follows.

- There are $t+1$ hospitals and $e+t$ patients. Each hospital $j\in[t]$ corresponds to a vertex $j\in V$ and hospital $t+1$ corresponds to a dummy hospital. Each *edge-type* patient $i\in[e]$ corresponds to an edge $(u_i,v_i)\in E$ and each *vertex-type* patient $i\in[e+t]\setminus[e]$ corresponds to a vertex $i-e\in V$.
- For each hospital $j\in[t]$, $c_j=1$; and for hospital $t+1$, $c_{t+1}=0$.
- For each edge-type patient $i\in[e]$, $v_{iu_i}=v_{iv_i}=t^2$ and $v_{ij}=0$ for any other hospital $j\in[t+1]\setminus\{u_i,v_i\}$. That is, i only wants hospitals corresponding to the vertices of its edge.

- For each vertex-type patient $i \in [e+t] \setminus [e]$, $v_{i(i-e)} = 1$ and $v_{ij} = 0$ for any other hospital $j \in [t+1] \setminus \{i-e\}$. That is, i only wants the hospital corresponding to its own vertex.

- Finally, $B = e + k$ and $T = et^2 + k$.

It is easy to see that the reduction takes polynomial time and produces an instance of *DPaW* where all the parameters are polynomial in the size of (G, k). We have the following two lemmas.

Lemma 5.2. $(G, k) \in VC \Rightarrow (c, v, B, T) \in DPaW$.

Proof. Letting V' be a vertex cover of G with $|V'| = k$, we construct an assignment $A = (a, w)$ as follows.

- $w_j = 0$ for each hospital $j \in V' \cup \{t+1\}$ and $w_j = t^2$ for each hospital $j \in V \setminus V'$.

- For each vertex-type patient $i \in [e+t] \setminus [e]$, if $i - e \in V'$ then $a(i) = i - e$, otherwise $a(i) = t + 1$.

- For each edge-type patient $i \in [e]$, if $u_i \in V'$ then $a(i) = u_i$; otherwise $a(i) = v_i$ (and $v_i \in V'$).

It is easy to see that the construction of A takes polynomial time. To see why A is budget-feasible, notice that there are exactly k vertex-type patients and e edge-type patients served in hospitals $\{1, \ldots, t\}$, with cost 1 each, thus the total cost is $e + k = B$.

Now we show that A is stable. For each edge-type patient $i \in [e]$, by the definition of a vertex cover we have $a(i) \in V'$, thus $w_{a(i)} = 0$ and $v_{ia(i)} - w_{a(i)} = t^2$, which is the maximum utility i can get from any hospital. For each vertex-type patient $i \in [e+t] \setminus [e]$ such that $i - e \in V'$, we have $v_{ia(i)} - w_{a(i)} = v_{i(i-e)} - w_{i-e} = 1 - 0 = 1$, which is again the maximum utility i can get from any hospital. Moreover, for each vertex-type patient i such that $i - e \notin V'$, we have $w_{i-e} = t^2$, $v_{i(i-e)} - w_{i-e} = 1 - t^2 < 0$, and $v_{ia(i)} - w_{a(i)} = v_{i(t+1)} - w_{t+1} = 0 \geq v_{ij} - w_j$ for each $j \in [t+1]$. Accordingly, the assignment A is stable.

Finally, the social welfare of A is

$$\begin{aligned} SW(A) &= \sum_{i\in[e+t]} v_{ia(i)} - w_{a(i)} = \sum_{i\in[e]} v_{ia(i)} - w_{a(i)} + \sum_{i\in[e+t]\setminus[e]} v_{ia(i)} - w_{a(i)} \\ &= et^2 + \sum_{i\in[e+t]\setminus[e],\ i-e\in V'} 1 = et^2 + |V'| = et^2 + k = T. \end{aligned}$$

In sum, $(c, v, B, T) \in DPaW$ as desired. □

Lemma 5.3. *$(c, v, B, T) \in DPaW \Rightarrow (G, k) \in VC$.*

Proof. Letting $A = (a, w)$ be the optimal stable budget-feasible assignment, we have $SW(A) \geq T = et^2 + k$. Letting V' be the set of vertices whose corresponding hospitals have waiting time 0 in A, we show that V' is a vertex cover of G with size k.

First of all, since A is budget-feasible and each hospital in $[t]$ has cost 1, there can be at most $e + k$ patients served at these hospitals. Second, notice that each edge-type patient values a hospital for at most t^2 and each vertex-type patient values a hospital for at most 1. In order to achieve $SW(A) \geq et^2+k$, it must be the case that all e edge-type patients and exactly k vertex-type patients are served by hospitals in $[t]$: if there are $x < e$ edge-type patients and y vertex-type patients served by hospitals in $[t]$, then the social welfare can be at most $xt^2+y \leq xt^2+t \leq (e-1)t^2+t^2 = et^2 < et^2+k$, a contradiction. Moreover, each of the k vertex-type patient is served at a hospital he values for 1, each edge-type patient is served at a hospital which he values for t^2, and all such hospitals have waiting time 0 in A. Accordingly, these hospitals are all in V'. By construction, each edge has a vertex in V' and V' is a vertex cover of G. Since each vertex-type patient corresponds to a different vertex, $|V'| \geq k$.

Finally, it is easy to see that $|V'|$ cannot be larger than k: otherwise, since A is stable, all the $|V'|$ corresponding vertex-type patients are served by hospitals in V' and the total cost is $e + |V'| > e + k = B$, a contradiction. Therefore we have $|V'| = k$ and $(G, k) \in VC$ as desired. □

Theorem 5.1 follows directly from Lemmas 5.2 and 5.3. □

6 Ongoing and future work

Not every new problem offers the scope for a plethora of interesting avenues to explore but this is not the case with PaW. PaW is incredibly tweakable; with each tweak resulting in an interesting problem in its own right. In this section, we discuss a few variations of PaW as a starting point for anyone looking for interesting problems to work on.

6.1 PaW Monotone

Consider the case where m hospitals can be arranged in an increasing or decreasing(montone) order by cost. For any patient P_i, the patients values towards the hospitals are $v_{i1} \geq v_{i2} \geq v_{i3} \geq ... \geq v_{im}$(or $v_{i1} \leq v_{i2} \leq v_{i3} \leq ... \leq v_{im}$), where each patient values the hospital with the highest cost the most and the monotonicity follows from there.

Practically, it makes sense to assume that the care offered in costlier hospitals might be better and as a result, patients prefer to be treated in higher cost hospitals. Theoretically, this case is lies somewhere between the general version of *PaW*[2] and the *PaW with common preferences*[3]. It offers a good enough constraint from the general PaW to make us believe that it might be easier. At the same time, it is a more general version of PaW with common preferences which makes us believe that it should be harder. Since PaW is strongly NP-hard and PaW with common preferences has an FPTAS, it would be interesting to know where the *PaW Monotone* version falls.

PaW Monotone has proven elusive to attempts at pseudo-polynomial reductions so far. An important reason is that the monotonicity of the values offers difficulty in reductions either by way of structure(difficult to encode graphs and ensure monotonicity at the same time) or by leading to an exponential blow up of values during reductions. On the other hand, *PaW Monotone* being a more general version does not give the properties of an ordered assignment and a neat structural characterisation of waiting times that *PaW with common preferences* offers.

6.2 PaW Two piece linear

The *PaW Two piece linear* is closely related to *PaW with common preferences*[3] and differs from it in the following way. Instead of each patient P_i having a single value for getting the medical treatment, every patient has two values(say $v_{i(high)}, v_{i(low)}$), one for each set of hospitals. There exists a hospital H_k, where the the patients value treatments at all hospitals on or above H_k favourably and unfavourably for all hospitals on or below H_k. Every patient P_i values the hospitals in the following manner, $v_{i(high)}q_1$,$v_{i(high)}q_2$, $v_{i(high)}q_3$, ...$v_{i(high)}q_k$,$v_{i(low)}q_{k+1}$,$v_{i(low)}q_{k+2}$,.. $v_{i(low)}q_m$. Here, every patient values treatments at hospitals after hospital H_k unfavourably. It is important to note that, the $v_{i(high)}$ and $v_{i(low)}$ of different patients are not related but it is necessary that, $v_{i(high)} \gg v_{i(low)}$ for each patient.

6.3 PaW Piece wise linear

The *PaW Piece wise linear* is a more general version of the *PaW Two piece linear* case. Here, instead of dividing the hospital into two(high, low) categories, we divide the hospitals into p categories, where each patient values all the hospitals in a particular category the same way.

On first glance, *PaW Two piece linear* and *PaW Piece wise linear* might appear easy to solve as it is a matter of allocating patients to each category and then, using the FPTAS for *PaW with common preferences*[3] to allocate hospitals for patients within a category. Allocating patients to different categories becomes challenging as the structure of waiting times and the ordering within groups has not yet been found.

References

[1] Yoram Barzel. A theory of rationing by waiting. *The Journal of Law Economics*, 17(1):73–95, 1974.

[2] M. Braverman, J. Chen, and S. Kannan. Optimal provision-after-wait in healthcare. *Mathematics of Operations Research*, 41(1):352–376, 2016.

[3] H. Chan and J. Chen. Provision-after-wait with common preferences. In *15th International Conference on Autonomous Agents and Multiagent Systems (AAMAS'16),* to appear, 2016.

[4] Stefan Felder. To wait or to pay for medical treatment? restraining ex-post moral hazard in health insurance. *Journal of Health Economics*, 27(6):1418 – 1422, 2008.

[5] M. R. Garey and D. S. Johnson. " strong " np-completeness results: Motivation, examples, and implications. *J. ACM*, 25(3):499–508, July 1978.

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

[7] Venkatesan Guruswami, Jason D. Hartline, Anna R. Karlin, David Kempe, Claire Kenyon, and Frank McSherry. On profit-maximizing envy-free pricing. In *Proceedings of the Sixteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '05, pages 1164–1173, Philadelphia, PA, USA, 2005. Society for Industrial and Applied Mathematics.

[8] Brian Hayes. The Easiest Hard Problem, 2002.