Constructing Treatment Portfolios Using Affinity Propagation

Delbert Dueck[1], Brendan J. Frey[1,2], Nebojsa Jojic[3], Vladimir Jojic[1,3,4], Guri Giaever[2], Andrew Emili[2], Gabe Musso[2], and Robert Hegele[5]

[1] Electrical and Computer Engineering, University of Toronto, Canada
[2] Center for Cellular and Biomolecular Research, University of Toronto, Canada
[3] Machine Learning and Statistics, Microsoft Research, Redmond, USA
[4] Computer Science, Stanford University, USA
[5] Cardiovascular Genetics Laboratory, Robarts Research Institute, London, Canada

Abstract. A key problem of interest to biologists and medical researchers is the selection of a subset of queries or treatments that provide maximum utility for a population of targets. For example, when studying how gene deletion mutants respond to each of thousands of drugs, it is desirable to identify a small subset of genes that nearly uniquely define a drug 'footprint' that provides maximum predictability about the organism's response to the drugs. As another example, when designing a cocktail of HIV genome sequences to be used as a vaccine, it is desirable to identify a small number of sequences that provide maximum immunological protection to a specified population of recipients. We refer to this task as 'treatment portfolio design' and formalize it as a facility location problem. Finding a treatment portfolio is NP-hard in the size of portfolio and number of targets, but a variety of greedy algorithms can be applied. We introduce a new algorithm for treatment portfolio design based on similar insights that made the recently-published affinity propagation algorithm work quite well for clustering tasks. We demonstrate this method using the two problems described above: selecting a subset of yeast genes that act as a drug-response footprint, and selecting a subset of vaccine sequences that provide maximum epitope coverage for an HIV genome population.

1 Treatment Portfolio Design (TPD)

A central question for any computational research collaborating with a biologist or medical researcher is in what form computational analyses should be handed over to the experimentalist or clinician. While application-specific predictions are often most appropriate, we have found that in many cases what is needed is a selection of potential options available to the biologist/medical researcher, so as to maximize the amount of information gleaned from an experiment, which often can be viewed as consisting of independently assayed targets. If the number of options is not too large, these can be discussed and selected by hand. On the other hand, if the number of possibilities is large, a computational approach may be needed to select the appropriate options. This paper describes the framework and approaches that emerged while trying to address problems of this type with our collaborators. In particular, we show how the affinity propagation algorithm [1] can be used to effectively to approach this task.

For concreteness, we will refer to the possible set of options as ‘treatments’ and the assays used to measure the suitability of the treatments as ‘targets’. Each treatment has a utility for each target and the goal of what we will refer to as treatment portfolio design (TPD) is to select a subset of treatments (the portfolio) so as to maximize the net utility of the targets. The terms ‘treatment’, ‘target’ and ‘utility’ can take on quite different meanings, depending on the application. Treatments might correspond to queries, probes or experimental procedures, while targets might correspond to disease conditions, genes or DNA binding events.

Example 1: The treatments are a set of potential yeast gene deletion strains used to query drug response, the targets are all $\sim$6000 yeast gene deletion strains, the utility is the number of gene-drug interactions in all strains that are predicted by the selected portfolio of strains.

Example 2: The treatments are a large set of potential vaccines derived from HIV genomes, the targets are a population of HIV epitopes likely to be present in a demographic with high infection risk, the utility is the level of immunological protection, *i.e.*, number of epitopes present in the selected portfolio of HIV vaccines.

Example 3: The treatments are a set of baseline demographic, anthropometric, biochemical and DNA SNP variables thought to be predictive of cardiovascular end-points and postulated to form a clinical set of risk factors, the targets are $\sim$4,000,000 disease end-point targets comprising $\sim$20,000 patients and $\sim$200 conditions, the utility is the predictability of disease end-points, including risk.

Example 4: The treatments are a set of laboratory procedures used to synthesize biologically active compounds, the targets are a list of desired compounds to be synthesized, the utility is the negative financial cost needed to synthesize all target compounds using the selected portfolio of laboratory procedures.

Example 5: The treatments are a large set of microRNAs potentially involved in regulating the expression of disease-associated genes, the targets are a list of gene-disease pairs, the utility is the net corrected correlation between gene expression and expression of microRNAs in portfolio for all disease conditions.

The input to TPD is a set of potential treatments or queries $\mathcal{T}$, a representative population of targets $\mathcal{R}$ and a utility function $u : \mathcal{T} \times \mathcal{R} \to \mathbb{R}$, where $u(T, R)$ is the utility of applying treatment $T \in \mathcal{T}$ to target $R \in \mathcal{R}$. This utility may be based on a variety of factors, including the benefit of the treatment, the cost, the time to application, the time to response, the estimated risk, *etc.* The goal of computational TPD is to select a subset of treatments $\mathcal{P} \subseteq \mathcal{T}$ (called the ‘portfolio’) so as to maximize their net utility for the target population. A defining aspect of the utility function is that it is additive; for portfolio $\mathcal{P}$, the net utility is

$$\sum\nolimits_{R\in\mathcal{R}} \max_{T\in\mathcal{P}} u(T, R).$$

To account for the fact that some treatments are preferable to others regardless of their efficacy for the targets (*e.g.*, different setup costs), we use a treatment-specific cost function $c : \mathcal{T} \to \mathbb{R}$. The net utility, including the treatment cost is

$$U(\mathcal{P}) = \sum\nolimits_{R\in\mathcal{R}} \max_{T\in\mathcal{P}} u(T, R) - \sum\nolimits_{T\in\mathcal{P}} c(T). \tag{1}$$

Provided with $\mathcal{T}$, $\mathcal{R}$, u and c, the computational task is to find $\max_{\mathcal{P} \subseteq \mathcal{T}} U(\mathcal{P})$. Note that the number of treatments in the portfolio will be determined by balancing the utility with the treatment cost.

1.1 Relationship to K-Medians Clustering and Facility Location

Under certain conditions, TPD can be viewed as a K-medians clustering problem or facility location problem (a.k.a. the p-median model) [3, 4, 5]. Given a set of points $\mathcal{X}$ and a pairwise distance measure $d : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$, the goal of K-medians clustering is to select a subset of points as centers and assign every other point to its nearest center, so as to minimize the sum of distances. To control the number of identified centers, either the number of centers is pre-specified or a cost is associated with each center. Because of the combinatorics involved in selecting K centers, the K-medians clustering problem is NP-hard (c.f. [5]). TPD can be viewed as K-medians clustering, if the treatment set equals the target set ($\mathcal{T} = \mathcal{R}$). The application of K-medians clustering algorithms to TPD is discussed below.

In general, the treatment set does not equal the target set. Then, TPD can be viewed as a facility location problem, which is framed as opening up facilities or warehouses to service customers. Given a set of facilities that can potentially be opened $\mathcal{F}$, a set of customers $\mathcal{C}$, a distance function $d : \mathcal{C} \times \mathcal{F} \to \mathbb{R}$ and a facility opening cost function $c : \mathcal{F} \to \mathbb{R}$, the facility location problem [5] consists of identifying a subset of facilities and assigning every customer to a facility so as to minimize the net facility opening cost and distance of customers to facilities. Because the number of possible combinations of facilities to choose from is exponential in the number of potential and chosen facilities, the facility location problem is NP-hard.

Most work on approximation algorithms for K-medians clustering and facility location relies on d being metric (convex) (c.f. [5] for a review), but this is not necessary. Both problems can be formulated as binary-valued integer programs and then relaxed to linear programs. If the linear program solution is non-integer, it can be rounded, giving rise to various approximations. Unfortunately, these approximation algorithms are of limited practical value. We have experimented extensively with a data set of 400 Euclidean points derived from images of faces and found that CPLEX 7.1 takes several hours and gigabytes of memory to find solutions that can be found in less than one minute using a couple megabytes of memory via the affinity propagation algorithm [1] (data available at www.psi.toronto.edu/affinitypropagation).

2 Standard Algorithms Adapted to TPD

A variety of non-iterative and iterative algorithms for optimizing the objective functions can be formulated in a straight-forward fashion. A simple method is to start with an empty portfolio and add the single treatment T_1 that maximizes the net utility (including the treatment cost). All targets are assigned to that treatment and the current utility $v(R)$ for target R is set to $u(T_1, R)$. Next, another treatment is added to the portfolio and this treatment is chosen so as to maximize the net utility. This involves examining every treatment T not currently in the portfolio, computing a net utility gain

$\sum_{R \in \mathcal{R}} \max(u(T,R) - v(R), 0) - c(T)$, and selecting the treatment with maximum utility gain. This treatment, denoted T_2, is added to the portfolio. This procedure is repeated until another treatment cannot be added to the portfolio without decreasing the net utility. Note that in the absence of a treatment cost (*i.e.*, $c = 0$, assuming that all utilities are non-negative), all treatments would be added to the portfolio. Then, the number of treatments K may instead be specified, in which case the algorithm is terminated when K treatments have been added to the portfolio.

```
ALGORITHM 1: $K$-TREATMENTS CLUSTERING
Input: $\mathcal{T}, \mathcal{R}, u, c, K, M$
Repeat $M$ times:
    $\mathcal{P}' \leftarrow$ random subset of $\mathcal{T}$ of size $K$
    $\forall R \in \mathcal{R},\ \ p(R) \leftarrow \text{argmax}_{T \in \mathcal{P}'} u(T,R)$
    Repeat until convergence:
        Select $T \in \mathcal{P}'$ (in order)
        $\forall T' \in \mathcal{T} \setminus \mathcal{P}'$, compute
            $g(T') \leftarrow \sum_{R:p(R)=T} u(T',R) - u(T,R) - c(T') + c(T)$
        $T^{\text{alt}} \leftarrow \text{argmax}_{T' \in \mathcal{T} \setminus \mathcal{P}'} g(T')$
        If $g(T^{\text{alt}}) > 0$ then set $\mathcal{P}' \leftarrow (\mathcal{P}' \setminus \{T\}) \cup \{T^{\text{alt}}\}$
        $\forall R \in \mathcal{R},\ \ p(R) \leftarrow \text{argmax}_{T \in \mathcal{P}'} u(T,R)$
    If first repetition, then $\mathcal{P} \leftarrow \mathcal{P}'$; else if $U(\mathcal{P}') > U(\mathcal{P})$, then $\mathcal{P} \leftarrow \mathcal{P}'$
Output: $\mathcal{P}$
```

One problem with the above method is that after the subsequent addition of a treatment, previously-added treatments may no longer be the ones that maximize the utility. A natural extension is to initially find K treatments and then iteratively revisit treatments and consider replacing them with other treatments not in the current portfolio, until the portfolio converges. Alternatively, instead of deterministically initializing the portfolio, it can be randomly initialized to K treatments and then iteratively refined. The advantage of this approach is that a large number, M, of random initializations can be tried and the refined portfolio with highest net utility can be selected. To reflect the similarity of this approach to the standard K-means clustering and K-medians clustering algorithms [3], we will refer to it as 'K-treatments clustering'. Note that K-treatments clustering is not identical to K-means clustering or K-medians clustering, because treatments are neither means nor medians – in fact, they generally lie in a different space than the targets that are assigned to them. See Alg. 1 for details.

3 Modified Affinity Propagation for TPD

The recently-introduced affinity propagation algorithm is an exemplar-based clustering algorithm that operates by exchanging messages between data points until a subset of data points emerge as the cluster centers (exemplars) [1]. Unlike most other clustering algorithms, affinity propagation does not store and refine a fixed number of potential cluster centers, but instead simultaneously considers all data points as potential cluster centers. Data points exchange two kinds of message: the responsibility sent from point i to point k indicates how well-suited point k is as the exemplar for point i in contrast to other potential exemplars; the availability sent from point k to point i indicates how much support point k has received from other points for being an exemplar. As the message-passing procedure proceeds, responsibilities and availabilities become more extreme until a clear set of exemplars and clusters emerges.

In [1], affinity propagation was shown to find better solutions than other frequently-used methods, including K-medians (K-centers) clustering and hierarchical agglomerative clustering. It should be kept in mind that for small problems, *e.g.* < 500 points, linear programming [5] can often be used to find an exact solution. Also, when the number of sought-after exemplars is quite low (*e.g.*, < 10), methods that use random initialization (*e.g.*, K-centers clustering) can work quite well. One randomly-initialized method that works quite well is the vertex substitution heuristic. Recently in [2], affinity propagation was shown to be significantly faster than the vertex substitution heuristic for moderately large problems. For example, 20 randomly-initialized runs of the vertex substitution heuristic took $\sim$10 days to find 454 clusters in 17,770 Netflix movies, whereas affinity propagation took $\sim$2 hours and achieved lower error.

The input to the affinity propagation algorithm is a set of similarities $\{s(i,k)\}$, where $s(i,k)$ is the similarity of point i to k, and a set of preferences $\{p(k)\}$, where $p(k)$ is the *a priori* preference that point k be chosen as an exemplar. After exchanging messages, affinity propagation identifies a set of exemplars $\mathcal{K}$ so as to maximize the net similarity $\sum_{i \notin \mathcal{K}} \max_{k \in \mathcal{K}} s(i,k) + \sum_{k \in \mathcal{K}} p(k)$. Viewing treatments as potential exemplars, we can adapt affinity propagation to TPD: if point i is a target and point k is a treatment, we can set $s(i,k)$ to the utility of that treatment for that target; if point k is a treatment, we can set $p(k)$ to the negative cost for that treatment.

However, one important difference between the problem statements for exemplar-based clustering and TPD is the distinction between treatments and targets. The original affinity propagation algorithm treats all points as potential exemplars and every non-exemplar point must be assigned to an exemplar. In TPD, only treatments can be selected as exemplars, and only targets have utilities for being assigned to exemplars (treatments). Treatments that are not selected for the portfolio (exemplar set) are neither exemplars nor assigned to another exemplar (treatment).

To allow some treatments to not be selected for the portfolio and also not be assigned to any other points, we introduce a special 'garbage collector' point and set the similarities of treatments to this point to zero. So, unless there is a net benefit in utility minus cost when including a treatment in the portfolio (exemplar set), it will be assigned to the garbage collector point. In summary, the following similarity constraints account for the *bipartite* structure of TPD:

$$\begin{gathered}
s(\text{target}, \text{treatment}) = u(\text{treatment}, \text{target}) \text{ and } s(\text{target}, \text{target}') = s(\text{target}, \text{garbage}) = -\infty \\
s(\text{treatment}, \text{garbage}) = 0 \text{ and } s(\text{treatment}, \text{target}) = s(\text{treatment}, \text{treatment}') = -\infty \\
s(\text{garbage}, \text{target}) = s(\text{garbage}, \text{treatment}) = -\infty \\
p(\text{treatment}) = -c(\text{treatment}) \text{ and } p(\text{target}) = p(\text{garbage}) = \infty
\end{gathered}$$

The last constraints ensure that targets cannot be selected as exemplars and that the garbage collection point is always available as an exemplar.

It turns out that when the above constraints are inserted into the original affinity propagation updates, certain simplifications occur and the garbage collection point need not be explicitly represented. In fact, many messages need not be computed and the algorithm reduces to messages exchanged on a bipartite graph connecting treatments and targets. The resulting algorithm is provided in Alg. 2 – note that messages need only be exchanged between a treatment and target if the utility is not $-\infty$. The input $\lambda \in (0, 1)$ is a damping factor that is used to improve convergence.

ALGORITHM 2: BIPARTITE AFFINITY PROPAGATION

```
Input: $\mathcal{T}, \mathcal{R}, u, c, \lambda$
Initialization: $\forall T, R : u(T, R) > -\infty$, set $a(T, R) \leftarrow 0,\ r(T, R) \leftarrow 0$
Repeat until convergence:
    Update responsibilities: $\forall T, R : u(T, R) > -\infty$, set
        $r(T,R) \leftarrow \lambda r(T,R) + (1-\lambda) \cdot \big(u(T,R) - \max_{\substack{T' \in \mathcal{T} \setminus \{T\}: \\ u(T',R) > -\infty}} \{u(T',R) + a(T',R)\}\big)$   (2)
    Update availabilities: $\forall T, R : u(T, R) > -\infty$, set
        $a(T,R) \leftarrow \lambda a(T,R) + (1-\lambda) \cdot \min\{0, \sum_{\substack{R' \in \mathcal{R} \setminus \{R\}: \\ u(T,R') > -\infty}} \max(0, r(T,R')) - c(T)\}$   (3)
Output:
    $\mathcal{P} \leftarrow \{T : \sum_{\substack{R \in \mathcal{R}: \\ u(T,R) > -\infty}} \max(0, r(T,R)) > c(T)\}$   (4)
```

This algorithm can be derived as an instance of the max-product algorithm in a factor graph describing Eq. 1. Here, we provide some intuition for why the updates make sense. Initially, the availabilities $a(\cdot, \cdot)$ are zero and Eq. 2 indicates that the responsibility of treatment T for target R is set to its utility minus the largest competing utility of another treatment for the same target. In subsequent iterations, the utilities of competing treatments are modulated by their availabilities. As indicated in Eq. 3, the availability of treatment T for target R is set to the sum of its responsibilities for other treatments minus its cost. Only positive responsibilities are included in this sum, because for a treatment to be deemed useful, it is only necessary that some targets yield high utility, not that all targets yield high utility. The availability is not allowed to rise above zero; this acts to prevent a treatment that accounts for a large number of targets from dominating other potential treatments. After convergence, Eq. 4 compares the net responsibility of target T with its cost, and includes it in the portfolio if the benefit outweighs the cost.

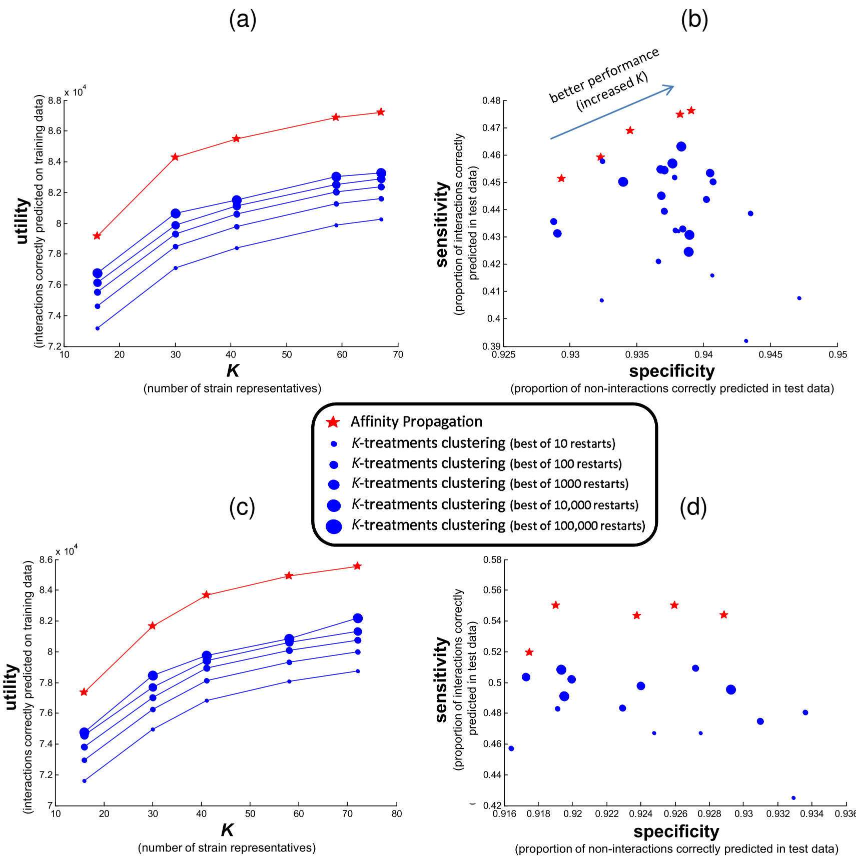

Fig. 1. Performance with affinity propagation and K-treatments clustering on finding yeast strains that are representative of gene-drug interaction profiles. (a) and (b) show training and test results, for a held-out test set of randomly-selected drugs; (c) and (d) show similar results, but where the held-out test drugs consisted of a small number of drugs with different dosage levels and exposure times. (a) and (c) show that affinity propagation maximizes the net utility of the training set better than the best of hundreds of thousands of random restarts of K-treatments clustering. (b) and (d) show that on the test drug set, affinity propagation has better sensitivity at a given specificity than K-treatments clustering.

4 Application 1: Selecting Yeast Gene Deletion Queries for Drug Profiling

We applied TPD to a data set showing the interaction of 1259 different drug-dosage-exposure combinations (called just drugs hereon in for brevity) with yeast gene-deletion strains. Our goal was to find a small query set of genes (a subset of the 5985 yeast genes) on which new drugs could be tested for the purpose of predicting interactions between the drug and genes not in the query set. The selection of a query set would avoid needing to test all genes on new drugs, which can be costly, especially when expanding tests, *e.g.*, to include proteomic profiling. In this application, the treatment set equals the target set ($\mathcal{T} = \mathcal{R}$); in the next section, we describe results on an application where the treatment set and target set are disjoint.

Gene-drug interactions were measured using TAG3 molecular barcode arrays [6]. z-scores were calculated by standardizing the rank-normalized intensity for each strain under the given drug against the intensity from a set of untreated controls. We imposed a threshold score of 5 for there to be an interaction between the strain and drug, which yielded roughly $150,000$ interactions of a possible 7.5 million combinations (2%). The utility of a potential query gene deletion strain (treatment) for representing another gene deletion strain (target) was set to the number of drug responses that were active for both. To test the predictive power of TPD, this utility was computed using a training set of drugs consisting of only 90% of the original; results are reported for both uniformly sampling the 10% test set of drugs and using a non-random test set consisting of all different dosage levels and exposure times for a smaller set of drugs (still 10% of the total number of conditions).

TPD was performed using both affinity propagation and K-treatments clustering (using a huge number of random restarts), with results shown in Fig. 1. All runs of affinity propagation took less than five hours whereas the many runs of K-treatments clustering took over 3 days. Results for the uniformly-sampled test set are shown in Fig. 1(a) and (b), whereas similar results are shown in Fig. 1(c) and (d) for the test set with fewer drugs with varying dosages and exposure times. Fig. 1(a) and (c) plot the net utility (total number of drug-gene interactions in the training set accounted for by the portfolio) versus the number of treatments. Affinity propagation finds better representative yeast strains than K-treatments clustering, for both training sets.

Additionally, Fig. 1(c) and (d) plot sensitivity vs. specificity curves for both algorithms, using the held-out test sets. At any given specificity (proportion of non-interactions correctly predicted in the test data), affinity propagation achieves a higher sensitivity (proportion of interactions correctly predicted in the test data) than K-treatments clustering. Note that as the size of the portfolio (K) increases, both specificity and sensitivity experience corresponding increases.

It is evident from Fig. 1 that many thousands of random restarts of K-treatments clustering are needed to find good solutions. For example, for all data points shown in both plots, we found that the portfolios found by affinity propagation never represented fewer than 7 target genes, whereas in all but one case for K-treatments clustering (the lowest, $K = 16$), the portfolios included singleton genes, leading to less-accurate predictions.

Exemplar genes found by both affinity propagation and the K-treatments algorithm were also analyzed for functional enrichment [7]. In nearly all cases, the list of exemplars was not over-enriched for any functional category, indicating that these representatives were well dispersed in terms of biological function.

5 Application 2: Selecting HIV Strains that Maximize Immune Target Coverage

Next, we pose the problem of HIV vaccine cocktail design as a TPD. The idea here is to find a set of optimal HIV strains for the purpose of priming the immune systems of many patients. The treatments $\mathcal{T}$ are thousands of HIV strain sequences (available at www.hiv.lanl.gov). The targets $\mathcal{R}$ are a set of short sequences (patches, fragments) that correspond to the epitopes that immune systems respond to (we use all nonamers or 9-mers). The utility $u(T,R)$ of a strain T for a fragment R would ideally be set to its potential for immunological protection, but following [8, 9, 10, 12] we set it to the frequency of the fragment in the database of HIV sequences, if fragment R is present in strain T and 0 otherwise. The net utility is also referred to as 'coverage'.[1]

Fig. 2 shows aligned pieces of Gag protein from several different strains, with two variable sites marked by arrows as well as known or predicted T-cell epitopes for the MHC molecules of five different patients taken from the WA cohort [11]. Epitopes recognizable by a single patient are shown in a single color, and each patient is assigned a different color. The white patient could be immunized against three forms of the same epitope: KKYKLKHIV, KKYQLKHIV, KKYRLKHIV. In this small example, we can design a vaccine consisting of the following segments which epitomizes in an immunological sense the seven strains shown in the figure: VLSGGKLDKWEKIRLRPGGKKKYK-LKHIVWASRELERF LSGGKLDRWEKIRLR KKKYQLKHIVW KKKYRLKHIVW.

A lot of discussion among HIV vaccine experts has been focused on the need for constraining vaccine constructs optimized for coverage to resemble naturally occurring strains [10, 9]. This is motivated by several pieces of evidence that deviation from naturally occurring strains often reduces efficacy in animal models as well as in vaccine trials, both in terms of the cellular and antibody responses. Thus, [9] proposes enrichment of the vaccine with a sequence that sits in the center of the HIV phylogenetic tree, so that this single native-like (but still artificially derived) strain is used to provide high coverage of immune targets in as natural way as possible, while the additional coverage is achieved with an epitome fragment or fragments. In contrast, in their recent paper [10] Fischer *et. al.* avoid the use of fragments altogether, and propose building the entire vaccine out of multiple strain-like constructs optimized by simulated strain recombination, dubbed 'mosaics'. A mosaic vaccine is therefore a cocktail of artificially-derived strains, not existent among the observed strains of the virus, but achievable by recombining many times the existing strains. These vaccine components resemble natural strains, but have higher nonamer coverage than what would be expected from a cocktail of natural strains. Mosaics can always achieve higher coverage

[1] Despite its simplicity, this problem set-up is quite biologically relevant. The immune system recognizes pathogens by short protein segments called epitopes. These targets are recognized both on the surface of the free viral particles, as well as on the surface of the infected cells, where the peptide targets are presented by the cell's own MHC molecules in charge of signaling about the normal or abnormal protein expression in the cell. Due to a need for efficiency, the immune system takes longer to recognize a new pathogen the first time it is encountered than in subsequent infections. To prime such an adaptive system against foreign intruders, various vaccination strategies have been developed, all essentially with the same goal — to expose the patient to a harmless vaccine that shares similarities with a targeted pathogen, so that in the immune system gets prepared for the true infection. For many pathogens, with HIV being a prime example, genetic diversity poses a significant problem for vaccine design. The goal of vaccine design is to load the vaccine with targets that would work for multiple strains and multiple patients [8, 9, 10, 12].

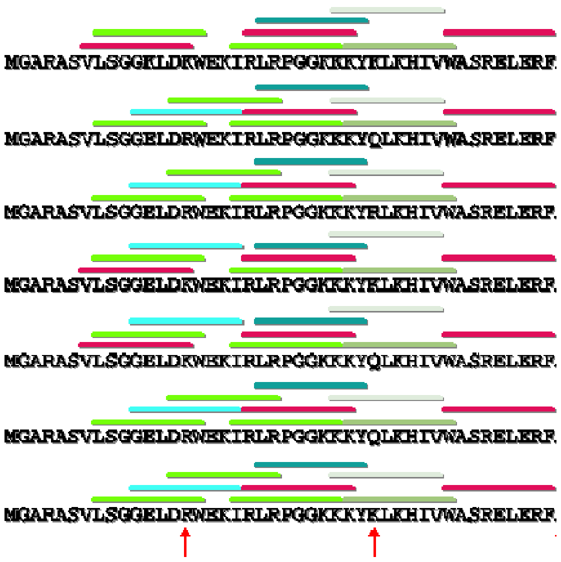
MGARASVLSGGKLDKWEKIRLRPGGKKKYKLKHIVWASRELERF
MGARASVLSGGELDRWEKIRLRPGGKKKYQLKHIVWASRELERF
MGARASVLSGGELDRWEKIRLRPGGKKKYRLKHIVWASRELERF
MGARASVLSGGELDKWEKIRLRPGGKKKYKLKHIVWASRELERF
MGARASVLSGGELDKWEKIRLRPGGKKKYQLKHIVWASRELERF
MGARASVLSGGELDRWEKIRLRPGGKKKYQLKHIVWASRELERF
MGARASVLSGGELDRWEKIRLRPGGKKKYKLKHIVWASRELERF

Fig. 2. Fragments of Gag protein with epitopes recognized by several HIV-infected patients. Epitopes recognizable by a single patient are shown in a single color; mutations marked by red arrows escape MHC I binding.

than natural strains, so while they may not be viable as vaccines, they provide an upper bound on potential coverage.

As the data set of known HIV sequences is constantly growing, the potential for achieving high coverage with a cocktail of true natural strains is growing as well. Newly discovered strains differ from existing ones mostly by the combination of previously seen mutations, rather than by the presence of completely new nonamers. In fact, Fischer *et. al.* have increased the Gag vaccine coverage by their use of mosaic by some 4–5% in comparison to natural strain cocktails for Gag and Nef protein vaccines consisting of 3–5 components. This is not much larger than the differences of around 2% of all nonamers that they report among coverage scores of mosaics optimized on different datasets (even within the same HIV clade). Furthermore, as the problem is NP-hard, the natural strain cocktails (treatment portfolios) in their paper are found by a greedy technique (a combination of K-treatments clustering and the vertex substitution heuristic), which may further decrease the perceived potential of natural strain cocktails, especially for a larger number of components. For a large M-clade dataset consisting of 1755 Gag proteins from the LANL database, a Gag sequence consisting of the best 4 natural strains we could find had only 3% lower coverage than the mosaic of the same size optimized on the same data (69% vs 66%); the differences in the Pol gene were even lower. Obviously, as the dataset grows, the computational burden for finding the optimal cocktail grows exponentially, as is the case for the general TPD problem.

Furthermore, while potentially important for the cellular arm of the immune system, a vaccine components' closeness to natural strains is even more important for properly presenting potential targets of the humoral (antibody) arm of the immune system. As opposed to the T-cell epitopes, antibody epitopes are found on the surface of the folded proteins. It has been shown that slight changes in the HIV Env protein can cause it to mis-fold, and so naturally occurring HIV strains are more likely to function properly than artificially derived Env proteins.

In our experiments, we solve the TPD problem for the Gag vaccine cocktail optimization for larger cocktails, where the coverage approaches 80% or more, and where exhaustive search is computationally out of the question. Instead, we used the affinity propagation algorithm described above, and compare its achieved utility (coverage) with that of the greedy method and the mosaic upper bound [10]. Table 1 summarizes our results on 1755 strains from M-clade (combination of all clades, and thus the most diverse).

Table 1. The utility ("epitope coverage") of vaccine portfolios found by affinity propagation and a greedy method, including an upper bound on utility (found using mosaics)

	Natural strains		Artificial mosaic strains
Problem	Affinity propagation	Greedy	(upper bound)
Gag, $K = 20$	77.54%	77.34%	80.84%
Gag, $K = 30$	80.92%	80.14%	82.74%
Gag, $K = 38$	82.13%	81.62%	83.64%
Gag, $K = 52$	84.19%	83.53%	84.83%

These results show that affinity propagation achieves higher coverage than the greedy method. Importantly, these results also suggest that the sacrifice in coverage necessary to satisfy the vaccine community's often-emphasized need for natural components, may in fact be bearable if large datasets and appropriate algorithms are used to optimize coverage.

6 Summary

We introduced a computation problem called 'treatment portfolio design', which is a key problem for biologists and medical researchers who need select a set of options useful for extracting maximum information or utility from a set of targets. This problem is equivalent to the non-metric K-medians problem or facility location problem, but while these problems are not new, practical algorithms that produce good solutions are still elusive. We showed how the recently-introduced affinity propagation algorithm can be modified to perform treatment portfolio design. We demonstrated a greedy algorithm for TPD and affinity propagation on the problem of identifying a yeast gene-deletion query set for the purpose of drug profiling and identifying strains of HIV that together maximize coverage of epitopes. Both methods were useful for identifying treatment sets, but we found that affinity propagation achieved significantly higher utility values and better test set performance (in terms of sensitivity and specificity), even when the greedy method was re-run hundreds of thousands of times using different random initializations.

References

1. Frey, B.J., Dueck, D.: Clustering by passing messages between data points. Science 315, 972–976 (2007)
2. Frey, B.J., Dueck, D.: Affinity propagation and the vertex substitution heuristic. Science (in press)
3. MacQueen, J.: Some methods for classification and analysis of multivariate observations. In: Proc. 5th Berkeley Symp. on Mathematical Statistics and Probability, pp. 281–297. Univ. of California Press (1967)
4. Balinksi, M.L.: On finding integer solutions to linear programs. In: Proc. IBM Scientific Computing Symp. on Combinatorial Problems, pp. 225–248 (1966)
5. Charikar, M., Guha, S., Tardos, A., Shmoys, D.B.: A constant-factor approximation algorithm for the k-median problem. J. Comp. and Sys. Sci. 65(1), 129–149 (2002)
6. Pierce, S.E., Fung, E.L., Jaramillo, D.F., Chu, A.M., Davis, R.W., Nislow, C., Giaever, G.: A unique and universal molecular barcode array. Nature Methods 3(8), 601–603 (2006)
7. Maere, S., Heymans, K., Kuiper, M.: BiNGO: a Cytoscape plugin to assess overrepresentation of gene ontology categories in biological networks. Bioinformatics 21, 3448–3449 (2005)
8. Jojic, N., Jojic, V., Frey, B., Meek, C., Heckerman, D.: Using epitomes to model genetic diversity: Rational design of HIV vaccine cocktails. NIPS 18, 587–594 (2005)
9. Nickle, D.C., et al.: Coping with Viral Diversity in HIV Vaccine Design. PLoS Computational Biology 3(4), e75 (2007)
10. Fischer, W., Perkins, S., et al.: Polyvalent vaccines for optimal coverage of potential T-cell epitopes in global HIV-1 variants. Nature Medicine 13, 100–106 (2006)
11. Mallal, S.: The Western Australian HIV Cohort Study, Perth, Australia. Journal of Acquired Immune Deficiency Syndromes and Human Retrovirology 17(Suppl. 1), 23–27 (1998)
12. Jojic, V.: Algorithms for rational vaccine design. Ph.D. Thesis, University of Toronto (2007)