Redistribution Systems and PRAM

Paul Cohen
Tomek Loboda

School of Computing and Information
University of Pittsburgh
Pittsburgh, PA, USA
prcohen@pitt.edu

March 23, 2020

Abstract

Redistribution systems iteratively redistribute mass between groups under the control of rules. PRAM is a framework for building redistribution systems. We discuss the relationships between redistribution systems, agent-based systems, compartmental models and Bayesian models. PRAM puts agent-based models on a sound probabilistic footing by reformulating them as redistribution systems. This provides a basis for integrating agent-based and probabilistic models. PRAM extends the themes of probabilistic relational models and lifted inference to incorporate dynamical models and simulation. We illustrate PRAM with an epidemiological example.

1 Introduction

Every Autumn, a new freshman class enters our university. Some drop out during the year, but most go on to become sophomores. Some get their general education requirements out of the way, others jump into their major areas of study. By the end of their first year, the incoming class is distributed among several groups. The dynamics of this distribution, month by month, year by year, depends on many factors and can be hard to analyze. Some students take a few classes in a major and decide that it isn’t what they’d hoped for. We can model this straightforwardly as a conditional probability of sticking with the major given one’s experiences in it. But if the major has limited capacity, then the number of students who stick with a major affects the number students who enter it. This is more difficult to model because the probabilities of transitions in and out of a major change over time. At our university and others there is an ongoing redistribution of students among academic, social, and other groups. New groups emerge: computer science students who are supported by the GI Bill, nursing students with minors in information science, and so on.

PRAM is a framework for building redistribution models and simulating their dynamics. In PRAM models, groups are defined by attributes and the dynamics of redistribution are generated by rules that probabilistically change attribute values. PRAM modelers specify these rules and some initial groups, but they need not anticipate all possible groups; PRAM generates groups automatically. PRAM grows and shrinks groups by redistributing their masses to other groups, some of which emerge during a PRAM simulation.

We built PRAM to unify several kinds of models in a single framework. PRAM incorporates aspects of compartmental models (e.g., [1]), agent-based models (ABMs, e.g., [5, 3]) and probabilistic relational models (PRMs; e.g., [2]). Simulation of PRAM models is a kind of lifted inference [1]. We suspect that all these kinds of models are fundamentally very similar [1]. PRAM seeks to clarify the probabilistic inference done by agent-based simulations as a first step toward integrating probabilistic and agent-based methods, enabling new capabilities such as automatic compilation of probabilistic models from simulation specifications, replacing or approximating expensive simulations with inexpensive probabilistic inference, and unifying ABMs with important methods such as causal inference.
2 An Example

Consider the spread of influenza in a population of students at two synthetic schools, *Adams* and *Berry*. To simplify the example, assume that flu spreads only at school. Many students at *Adams* have parental care during the day, so when they get sick they tend to recover at home. Most students at *Berry* lack parental care during the day, so sick students go to school. Students may be susceptible, exposed or recovered.

Figure 1: The left panel shows the proportions of students exposed to flu at the artificial *Adams* and *Berry* schools over 50 time steps. The right panel shows 23 the proportions of exposed students at 23 schools in Pittsburgh.

Although *Adams* and *Berry* are identical in all respects other than the availability of parental care, the dynamics of flu, as simulated by PRAM, are different at the schools. This is shown in the left side of Figure 1. The reasons are that the probability of contracting flu at school depends on proportion of people who have it, and 80% of *Berry* students go to school when they are sick, while 60% of *Adams* students stay home. Similar, dynamics are seen for 23 schools in Pittsburgh. In this case, we specified that the probability of going home when sick is 0.9 for a pre-schooler, 0.5 for a middle-schooler and .1 for a high-school student.

PRAM redistributes the student populations in these examples between several groups. There are susceptible, exposed and recovered groups; and these levels of flu status are crossed with location – home or school – and also with particular schools – *Adams* or *Berry* in the first example and 23 schools in the second. Indeed, in the second example, PRAM begins with 433 groups and generates 2064 more groups as it simulates the dynamics of flu within schools.

3 Elements of PRAM Models

PRAM models comprise entities and rules. At present, entities are groups or sites. Groups have counts that are redistributed among groups, and they have two kinds of attributes: unary features, \( F \), such as flu_status and sex, and binary relations, \( R \) such as has_location. Groups are related to sites and sites aggregate information about the groups to which they are related in the sense that the term is used in \( \{ \}. \) For example, a site might calculate the total mass of related groups that are exposed to flu. All forward relations between groups and sites, such as \( g_1 \text{.has\_school} = \text{Adams} \) relate one group to one site. Inverse relations relate one site to a set of groups. Thus, if \( g_1 \text{.has\_school} = \text{Adams} \) and \( g_2 \text{.has\_school} = \text{Adams} \), the inverse relation \( \text{Adams\_school\_of} \) returns \( \{g_1, g_2\} \). Inverse relations are important for answering queries such as “which groups attend \( g_1 \)’s school?” Formally this would be \( g_1 \text{.has\_school\_school\_of} \), which would return \( \{g_1, g_2\} \). By mapping over entities it is easy to answer queries such as “what is the proportion of students at \( g_1 \)’s school that has been exposed to flu?” In effect, PRAM implements a simple relational database.

Besides entities, PRAM models have rules that apply to groups. All rules have mutually exclusive conditions, and each condition is associated with a probability distribution over mutually exclusive and exhaustive conjunctive actions. Thus, a rule will return exactly one distribution of conjunctive actions or nothing at all if no condition is true. For an illustration, look at the mutually exclusive clauses of rule flu_progression in Figure 2, and particularly at the middle clause: It tests whether the group’s flu_status == e (exposed to flu) and it specifies a distribution over three conjunctive actions. The first, which has probability 0.2, is that the group recovers and becomes happy (i.e., change flu_status to r and change mood to happy). The remaining probability mass is divided between remaining exposed and becoming bored, with probability 0.5, and remaining exposed and becoming annoyed, with probability 0.3.
def rule_flu_progression (group):
    flu_status = group.get_feature('flu')
    location = objects_related_by(group,'has_location')
    infection_probability = location.proportion_located_here([('flu','e')])
    if flu_status == 's':
        return ((infection_probability,
                  ('change_feature','flu','e'),('change_feature','mood','annoyed')),
                ((1 - infection_probability),('change_feature','flu','s')))
    elif flu_status == 'e':
        return ((.2, ('change_feature','flu','r'),('change_feature','mood','happy')),
                (.5, ('change_feature','flu','e'),('change_feature','mood','bored')),
                (.3, ('change_feature','flu','e'),('change_feature','mood','annoyed')))
    else flu_status == 'r':
        return ((.9, ('change_feature','flu','r')),
                (.1, ('change_feature','flu','s')))

def rule_flu_location (group):
    ...
    if flu_status == 'e' and income == '1':
        return ((.1, ('change_relation','has_location',location,home)),
                (.9, ('change_relation','has_location',location,location)))
    elif flu_status == 'e' and income == 'm':
        return ((.6, ('change_relation','has_location',location,home)),
                (.4, ('change_relation','has_location',location,location)))
    else flu_status == 'r':
        return ((.8, ('change_relation','has_location',location,school)),
                (.2, ('change_relation','has_location',location,location)))

Figure 2: Two PRAM rules. Rule_flu_progression changes the flu_status and mood features of a group.
Rule_flu_location changes a group's has_location relation.
Next, consider the preamble of rule flu_progression, which queries the group's flu status, then finds the group's location, and then calls the method proportion_located_here to calculate the proportion of flu cases at the location. (Proportion_located_here sums the counts of groups at the location that have flu, then divides by the sum of the counts of all the groups at the location.) In the rule's first clause, this proportion serves as a probability of infection. It is evaluated anew whenever the rule is applied to a group. In this way, rules can test conditions that change over time. Finally, the third clause of the rule represents the transition from flu_status = r back to flu_status = s, whereupon re-exposure becomes possible.

In addition to changing groups' features, rules can also change relations such as has_location. The second rule in Figure 2 says, if a group is exposed to flu and is low-income then change the group's location from its current location to home with probability 0.1 and stay at location with probability 0.9. If, however, the group is exposed and is middle-income, then it will go home with probability 0.6 and stay put with probability 0.4. And if the group has recovered from flu, whatever its income level, then it will go back to school with probability 0.8.

4 Groups are defined by their attributes

PRAM groups are defined by their features and relations in the following sense: Let \( \mathcal{F} \) and \( \mathcal{R} \) be features and relations of group \( g \), and let \( n \) be the count of \( g \). For groups \( g_1 \) and \( g_2 \), if \( \mathcal{F}_i = \mathcal{F}_j \) and \( \mathcal{R}_i = \mathcal{R}_j \), then PRAM will merge \( g_i \) with \( g_j \) and give the result a count of \( n_i + n_j \). Conversely, if a rule specifies a distribution of \( k \) changes to \( \mathcal{F}_i \) (or \( \mathcal{R}_i \)) that have probabilities \( p_1, p_2, \ldots, p_k \), then PRAM will create \( k \) new groups with the specified changes to \( \mathcal{F}_i \) (or \( \mathcal{R}_i \)) and give them counts equal to \((p_1 \cdot n_1), (p_2 \cdot n_2), \ldots, (p_k \cdot n_k)\).

To illustrate, consider a PRAM system with just a single attribute, flu_status, which takes values s, e and r. Figure 3 illustrates how groups are created, split and merged, and how their counts change.

![Figure 3: How PRAM splits, merges and creates groups to redistribute group counts.](image)

Suppose PRAM starts with two groups \( g_1 \) and \( g_2 \) (denoted by double-lined boxes) with flu_status = s and flu_status = e, and counts \( n_1 \) and \( n_2 \), respectively. A rule specifies that susceptible people become exposed with probability \( p \), so PRAM generates two potential groups (denoted by dotted lines) and redistributes the count of \( g_1 \) between them in proportions \( p, 1-p \). As groups in this simple example are defined by a single feature, these potential groups are identical with \( g_2 \) and \( g_1 \), respectively, so PRAM will redistribute \( n_1 \) to \( g_2 \) and \( g_1 \). Redistribution means that the entire count of a group, \( n_1 \) in this case, is distributed, so \( g_1 \)’s new count will be \( n_1 \cdot (1 \{ p \}) \) while the count of \( g_2 \) will be incremented by \( n_1 \cdot p \). However, something similar is going on with \( g_2 \): A rule specifies that some exposed people will recover with probability \( q \), so PRAM spawns two potential groups with counts of \( n_2 \cdot q \) and \( n_2 \cdot (1 \{ q \}) \), and distributes the first to the new recovered group and the second back to \( g_2 \). Finally, because the potential group labeled r doesn’t already exist, PRAM makes it a real group (with a solid line) and gives it the name \( g_{2,1} \), denoting that it is the first real group created by the action of rules on group \( g_2 \). After all this, the counts for the groups are:

\[
\begin{align*}
g_1 &: n_1 \cdot (1 \{ p \}) \\
g_2 &: n_1 \cdot p + n_2 \cdot (1 \{ q \}) \\
g_{2,1} &: n_2 \cdot q
\end{align*}
\]

Clearly, the system in Figure 3 can be iterated with these counts as a new starting point. Repeated iterations will yield the dynamics of group counts.

PRAM isn’t necessary for this simple example, which mirrors the SIR compartmental developed by Kermack and McKendrick in 1927 and is well understood [4]. However, PRAM handles vastly more complicated models.
allowing more features, more groups, relations between groups, multiple rules applying simultaneously to groups, and nonstationary probabilities. PRAM guarantees that group counts always obey the probabilities associated with rules, and that the order of rules and clauses within rules, and the order of application of rules to groups, have no effects on counts.

5 The PRAM Engine: Redistributing Group Counts

The primary function of the PRAM engine is to redistribute group counts among groups, as directed by rules, merging and creating groups as needed, in a probabilistically sound way. To illustrate the details of how PRAM redistributes counts, suppose a PRAM model starts with just the two rules in Figure 2 and two extant groups:

| name | flu | mood | location   | count |
|------|-----|------|------------|-------|
| g₁   | s   | happy| adams      | 900   |
| g₂   | e   | annoyed| adams    | 100   |

The features for these groups are $\mathcal{F}_1 = \{\text{flu} = s, \text{mood} = \text{happy}\}$ and $\mathcal{F}_2 = \{\text{flu} = e, \text{mood} = \text{annoyed}\}$, and both groups have the same relation: $\mathcal{R}_1 = \mathcal{R}_2 = \{\text{has\_school\_adams}\}$.

Redistribution Step 1: Generate Potential Groups When rule $\text{flu\_progression}$ is applied to $g₁$, it calculates the infection probability at $\text{adams}$ to be $100/(100 + 900) = .1$. $g₁$ triggers the first clause in the rule because $g₁\text{\_flu\_status} = = s$. So the rule specifies that the $\text{flu\_status}$ of $g₁$ changes to $e$ with probability 0.1 and changes to $s$ with probability 0.9. PRAM then creates two potential groups:

| name | flu | mood | location | count |
|------|-----|------|----------|-------|
| g₁₁  | e   | annoyed| adams    | 90    |
| g₁₂  | s   | happy | adams    | 810   |

These potential groups specify a redistribution of $n₁$, the count of $g₁$. We will see how PRAM processes redistributions, shortly.

Of the two rules described earlier, rule $\text{flu\_location}$ does not apply to $g₁$, but both apply to group $g₂$. When multiple rules apply to a group, PRAM creates the cartesian product of their distributions of actions and multiplies the associated probabilities accordingly, thereby enforcing the principle that rules' effects are independent. (If one wants dependent effects they should be specified within rules.) To illustrate, rule $\text{flu\_progression}$ specifies a distribution of three actions for groups like $g₂$ that have $\text{flu\_status} = e$, with associated probabilities 0.2, 0.5, 0.3; while rule $\text{flu\_location}$ specifies two locations for groups that have $\text{flu\_status} = e$ and $\text{flu\_status} = m$, with probabilities 0.6 and 0.4. Thus, for $g₂$, there are six joint actions of these two rules, thus six potential groups:

| name | flu | mood | location | count |
|------|-----|------|----------|-------|
| $g₂₁$ | r   | happy| home     | 100 · 0.2 · 0.6 = 12.0 |
| $g₂₂$ | r   | happy| adams    | 100 · 0.2 · 0.4 = 8.0   |
| $g₂₃$ | e   | bored| home     | 100 · 0.5 · 0.6 = 30.0  |
| $g₂₄$ | e   | bored| adams    | 100 · 0.5 · 0.4 = 20.0  |
| $g₂₅$ | e   | annoyed| home    | 100 · 0.3 · 0.6 = 18.0  |
| $g₂₆$ | e   | annoyed| adams    | 100 · 0.3 · 0.4 = 12.0  |

These groups redistribute the count of $g₂$ (which is 100) by multiplying it by the product of probabilities associated with each action.

Redistribution Step 2: Process Potential Groups PRAM applies all rules to all groups, collecting potential groups as it goes along. Only then does it redistribute counts, as follows:

1. Extant groups that spawn potential groups have their counts set to zero;
2. Potential groups that match extant groups (i.e., have identical $\mathcal{F}$s and $\mathcal{R}$s) contribute their counts to the extant groups and are discarded;
3. Potential groups that don’t match extant groups become extant groups with their given counts.

So: Extant groups $g₁$ and $g₂$ have their counts set to zero. Potential group $g₁₁$ has the same features and relations as $g₁$ so it contributes its count, 810, to $g₁$ and is discarded. Likewise, potential group $g₁₂$ matches $g₂$ so it contributes 90 to $g₂$ and is discarded. Potential group $g₂₁$ also matches $g₂$, so it contributes 12 to $g₂$ and is discarded, bringing $g₂$’s total to 102. Potential groups $g₂₂$, $g₂₃$, $g₂₄$, and $g₂₅$ do not match any extant group, so they become extant groups. The final redistribution of extant groups $g₁$ and $g₂$ is:
The second iteration produces $g_1 = 706.632$, $n_2 = 119.768$, $n_{2,1} = 26.4$, $n_{2,1,1} = 0.24$, $n_{2,2} = 25.6$, $n_{2,3} = 26.0$, $n_{2,4} = 24.4$, $n_{2,5} = 36.36$.
Another reason to prefer PRAM over conventional ABMs is that the probabilistic foundations and guarantees of ABMs are murky, at best. In agent-based models agents probabilistically change state. State can be represented as attribute values such as health status, monthly income, age, political orientation, location and so on. A population of agents has a joint state that is typically a joint distribution; for example, a population has a joint distribution over income levels and political beliefs. ABMs are a popular method for exploring the dynamics of joint states, which can be hard to estimate when attribute values depend on each other, and populations are heterogeneous in the sense that not everyone has the same distribution of attribute values, and the principal mechanism for changing attribute values is interactions between agents. ABMs are no doubt engines of probabilistic inference, but it is difficult to say anything about the models that underlie the inference. PRAM seeks to clarify the probabilistic inference done by agent-based simulations.

7 Future Work

PRAM code is available on github [6]. It has run on much larger problems, including a simulation of daily activities in Allegheny County that involved more than 200,000 groups. PRAM runtimes are proportional to \( n \) the number of groups, not the group counts, so PRAM can be much more efficient than agent-based simulations (ABS). Indeed, when group counts become one, PRAM is an ABS, but in applications where agents or groups are functionally identical PRAM is more efficient than ABS.

Because \( n \) depends on the numbers of features and relations, and the number of discrete values each can have, PRAM could generate enormous numbers of groups. In practice, the growth of \( n \) is controlled by the number of groups in the initial population and the actions of rules. Typically, \( n \) grows very quickly to a constant, after which PRAM merely redistributes counts between these groups. In the preceding example, the initial \( n = 8 \) groups grew to \( n = 44 \) on the first iteration and \( n = 52 \) on the second, after which no new groups were added.

This dependence between \( n \) and the actions of rules suggests a simple idea for compiling populations given rules: Any feature or relation that is not mentioned in a rule need not be in groups’ \( \mathcal{F} \) or \( \mathcal{A} \). Said differently, the only attributes that need to be in groups’ definitions are those that condition the actions of rules. Currently we are building a compiler for PRAM that automatically creates an initial set of groups from two sources: A database that provides \( \mathcal{F} \) and \( \mathcal{A} \) for individuals and a set of rules. The compiler eliminates from \( \mathcal{F} \) and \( \mathcal{A} \) those attributes that aren’t queried or changed by rules, thereby collapsing a population of individuals into groups with known counts.

Attributes with continuous values obviously can result in essentially infinite numbers of groups. (Imagine one group with a single real-valued feature and one rule that adds a standard normal variate to it. Such a PRAM model would double the number of groups on each iteration without limit.) Rather than ban real-valued attributes from PRAM we are working on a method by which groups have distributions of such attributes and rules change the parameters of these distributions. We are developing efficient methods by which PRAM generates new potential groups and tests whether they match extant groups. To illustrate the approach, suppose we have a population distribution of income which, for the sake of simplicity, is uniform over the range [0,100]. Suppose we define two groups according to this exogenous distribution \( g_{\text{low}} \) has income less than 50, whereas \( g_{\text{med}} \) has income greater than or equal to 50. Now suppose that every member of \( g_{\text{low}} \) gets a 20% percent raise. It turns out that 16% of \( g_{\text{low}} \) will make more than 50 after the raise. Let’s call this fraction the upwardly mobile, or \( \text{UM} \). Suppose \( g_{\text{low}} \) has count \( n_{\text{low}} = 100 \) and \( g_{\text{med}} \) has \( n_{\text{med}} = 500 \). If income is truly uniformly distributed in \( g_{\text{low}} \), then after a uniform 20% raise, 16% of \( g_{\text{low}} \) will make more than 50. If income were the only factor that defined groups, then PRAM redistribution should reduce the count of \( g_{\text{low}} \) by 16 and increase the count of \( g_{\text{med}} \) by 16, as illustrated in Figure 4.

This example suggests that groups can be defined based on real-valued features:

1. An exogenous distribution of income is defined and divided into two regions which we’ll call the low and medium income regions. In general, we will define a multivariate distribution and divide it into many regions.
2. Two groups are defined, each with an endogenous distribution of income; call these distributions \( I_{\text{low}} \) and \( I_{\text{med}} \). In general, groups are defined by the relationships between their (multivariate) endogenous distributions and the regions of the (multivariate) exogenous distribution. One kind of relationship is “contained in”: \( I_{\text{low}} \) and \( I_{\text{med}} \) are contained in the low income and medium income regions, respectively.
3. A labeling function sweeps the endogenous distribution of a group and returns the relationships that hold between elements of the distribution and regions. For example, before the raise, the labeling function would say that all the element of \( I_{\text{low}} \) are contained in the low income region.
4. A rule would change \( I_{\text{low}} \) by multiplying \( I_{\text{low}} \) by 1.2.
5. The labeling function would relabel \( I_{\text{low}} \). This time, it would label 84% of the distribution as contained in the low income region and 16% as contained in the medium income region.
6. PRAM would use these labels as features and so would split \( g_{\text{low}} \) into two groups: One would be re-merged with \( g_{\text{low}} \), the other would merge with \( g_{\text{med}} \).
Figure 4: The dividing line between low and medium income is 50. If $g_{low}$ gets a 20% raise, then 16% of $g_{low}$ will make more than 50. After redistribution, this fraction of $g_{low}$ increases the count of $g_{med}$. The count of $g_{low}$ is reduced accordingly.

We are currently working on this mechanism for groups that are defined as regions of multivariate distributions constructed automatically from databases.

In sum, while PRAM is a simple algorithm for redistributing counts of groups, it appears to unify several other modeling frameworks. The primary advantage of PRAM over ABS is that PRAM models are guaranteed to handle probabilities properly. The steps described in Section 5 ensure that group counts are consistent with the probability distributions in rules and are not influenced by the order in which rules are applied to groups, or the order in which rules’ conditions are evaluated. These guarantees are the first step toward a seamless unification of databases with probabilistic and PRAM models. The next steps, which we have already taken on a very small scale, are automatic compilation of probabilistic models given PRAM models, and automatic compilation of PRAM rules given probabilistic models. Probabilistic relational models, which inspired PRAM, integrate databases with lifted inference in Bayesian models; PRAM adds simulation to this productive mashup, enabling models of dynamics.

8 Acknowledgments

This work is funded by the DARPA program “Automating Scientific Knowledge Extraction (ASKE)” under Agreement HR00111990012 from the Army Research Office.

References

[1] Julie C. Blackwood and Lauren M. Childs. An introduction to compartmental modeling for the budding infectious disease modeler. Letters in Biomathematics, 5(1), pp.195-221. 2018. doi:10.1080/23737867.2018.1509026
[2] Lise Getoor, Ben Taskar (Eds.) Introduction to Statistical Relational Learning. 2007. MIT Press
[3] Grefenstette JJ, Brown ST, Rosenfeld R, Depasse J, Stone NT, Cooley PC, Wheaton WD, Fyshe A, Galloway DD, Sriram A, Guclu H, Abraham T, Burke DS. FRED (A Framework for Reconstructing Epidemic Dynamics): An open-source software system for modeling infectious diseases and control strategies using census-based populations. BMC Public Health, 2013 Oct;13(1), 940. doi: 10.1186/1471-2458-13-940.
[4] Kermack WO, McKendrick AG (August 1, 1927). “A Contribution to the Mathematical Theory of Epidemics”. Proceedings of the Royal Society A. 115 (772): 700-721. doi:10.1098/rspa.1927.0118
[5] Kalliopi Kravari and Nick Bassiliades. A Survey of Agent Platforms. 2015.
[6] Loboda, T. The version of PRAM reported here was developed by the first author. A better engineered version has been developed by Tomek Loboda: https://github.com/momacs/pram/ with documentation at https://github.com/momacs/pram/blob/master/docs/Milestone-3-Report.pdf. 2019