Probabilistic Relational Agent-based Models (PRAM)*

Paul R. Cohen and Tomasz D. Loboda

University of Pittsburgh, Pittsburgh, PA, USA {prcohen,tol7}@pitt.edu

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.

Keywords: agent-based model · probabilistic model · redistribution model · dynamical systems · simulation · modeling framework

1 Introduction

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 [7]. We suspect that all these kinds of models are fundamentally very similar.

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

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

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.

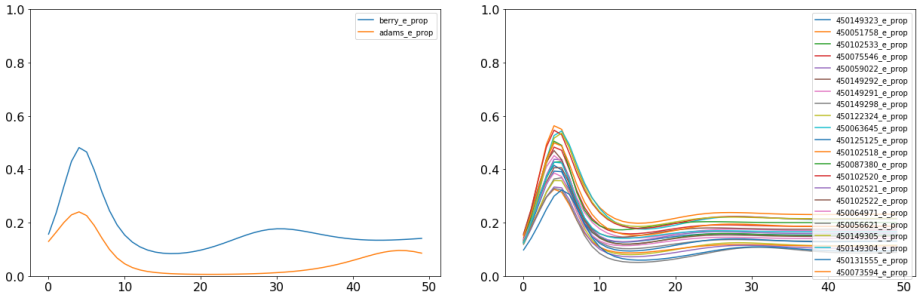

Fig. 1. The left panel shows the proportions of students exposed to flu at the artificial Adams and Berry schools over 50 time steps. The right panel shows 23 proportions of exposed students at 23 schools in Pittsburgh.

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 these schools, as 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 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*, $\mathcal{F}$, such as flu and sex, and binary *relations*, $\mathcal{R}$ 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 [2]. 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.has_school = Adams relate one group to one site. Inverse relations relate one site to a set of groups. Thus, if g_1.has_school = Adams and g_g.has_school = Adams, the inverse relation 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 `g1.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 school g_1 that has been exposed to flu?" In effect, PRAM implements a simple relational database.

```
rule_flu_progression():
    if group.feature.flu == 's':
        p_inf = n@_{feature.flu == 'e'} / n@  # n@ - count at the group's location
        dm     p_inf -> F:flu = 'e', F:mood = 'annoyed'
        nc 1 - p_inf
    if group.feature.flu == 'e':
        dm 0.2 -> F:flu = 'r', F:mood = 'happy'
        dm 0.5 -> F:flu = 'e', F:mood = 'bored'
        dm 0.3 -> F:flu = 'e', F:mood = 'annoyed'
    if group.feature.flu == 'r':
        dm 0.1 -> F:flu = 's'  # dm - distribute mass
        nc 0.9                 # nc - no change

rule_flu_location():
    if group.feature.flu == 'e':
        if group.feature.income == 'l':
            dm 0.1 -> R:@ = group.rel.home
            nc 0.9
        if group.feature.income == 'm':
            dm 0.6 -> R:@ = group.rel.home  # R:@ - location the group is at
            nc 0.4
    if group.feature.flu == 'r':
        dm 0.8 -> R:@ = group.rel.school
        nc 0.2
```

Fig. 2. Two PRAM rules. `rule_flu_progression` changes the `flu` and `mood` features of a group; `rule_flu_location` changes a group's `@` relation which denotes that group's current location. The distribute-mass operation is encode as `dm` and can set features `F` and relations `R` with a given probability.

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 is exposed to flu (i.e., `flu == e`) and it specifies a distribution over three conjunctive actions. For example, the first action, which has probability 0.2, is that the group recovers *and* becomes happy.

Next, consider the first clause, which calculates the proportion of flu cases at the group's location. This proportion subsequently serves as a probability

of infection and is evaluated anew whenever the rule is applied to a group. That is how rules can test conditions that change over time. Finally, the third clause represents the transition from recovered back to susceptible, whereupon re-exposure becomes possible.

In addition to changing groups' features, rules can also change relations. 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 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 i and j, if $\mathcal{F}_i = \mathcal{F}_j$ and $\mathcal{R}_i = \mathcal{R}_j$, then PRAM will merge i with 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 $(p_1 n_i), (p_2 n_i), \ldots, (p_k n_i)$.

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

Suppose PRAM starts with two groups $\mathtt{g}_1$ and $\mathtt{g}_2$ (denoted by double-lined boxes) with `flu=s` and `flu=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 $\mathtt{g}_1$ 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 $\mathtt{g}_2$ and $\mathtt{g}_1$, respectively, so PRAM will redistribute n_1 to $\mathtt{g}_2$ and $\mathtt{g}_1$. Redistribution means that the *entire* count of a group, n_1 in this case, is distributed, so $\mathtt{g}_1$'s new count will be $n_1(1-p)$ while the count of $\mathtt{g}_2$ will be incremented by $n_1 p$. However, something similar is going on with $\mathtt{g}_2$: A rule specifies that some exposed people will recover with probability q, so PRAM spawns two potential groups with counts of $n_2\, q$ and $n_2\,(1-q)$, and distributes the first to the new recovered group and the second back to $\mathtt{g}_2$. 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 $\mathtt{g}_{2\_1}$, denoting that it is the first real group created by the action of rules on group $\mathtt{g}_2$. After all this, the counts for the groups are:

$$\begin{aligned} \mathtt{g}_1 &: \; n_1(1-p) \\ \mathtt{g}_2 &: \; n_1 p + n_2(1-q) \\ \mathtt{g}_{2\_1} &: \; n_2 q \end{aligned}$$

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.

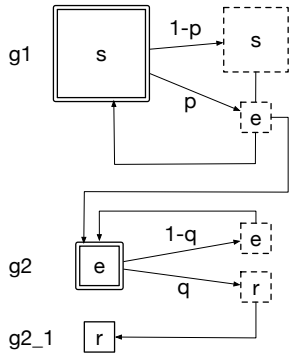

Fig. 3. How PRAM splits, merges and creates groups to redistribute group counts.

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 does that, suppose a PRAM model starts with just the two rules in Figure 2 and two *extant* groups:

name	flu	mood	location	count
g_1	s	happy	adams	900
g_2	e	annoyed	adams	100

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

The PRAM redistribution algorithm proceeds in three steps:

Redistribution Step 1: Generate Potential Groups. When `rule_flu_progression` is applied to g_1 it calculates the infection probability `p_inf` at `adams` to be 100/(100+900) = 0.1. g_1 triggers the first clause in the rule because `flu=s`. So the rule specifies that the g_1`.flu` 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
g_{1_1}	e	annoyed	adams	90
g_{1_2}	s	happy	adams	810

These potential groups specify a *redistribution* of n_1, the count of g_1. We will see how PRAM processes redistributions, shortly.

Of the two rules described earlier, `rule_flu_location` does not apply to g_1, but both apply to group g_2. 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 (dependent effects should be specified *within* rules). To illustrate, `rule_flu_progression` specifies a distribution of three actions for groups like g_2 that have `flu=e`, with associated probabilities $0.2, 0.5, 0.3$; while `rule_flu_location` specifies two locations for groups that have `flu=e` and `flu=m`, with probabilities 0.6 and 0.4. Thus, for g_2, there are six joint actions of these two rules, thus six potential groups:

name	flu	mood	location	count
g_{2_1}	r	happy	home	100 · 0.2 · 0.6 = 12.0
g_{2_2}	r	happy	adams	100 · 0.2 · 0.4 = 8.0
g_{2_3}	e	bored	home	100 · 0.5 · 0.6 = 30.0
g_{2_4}	e	bored	adams	100 · 0.5 · 0.4 = 20.0
g_{2_5}	e	annoyed	home	100 · 0.3 · 0.6 = 18.0
g_{2_6}	e	annoyed	adams	100 · 0.3 · 0.4 = 12.0

These groups redistribute the count of g_2 (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 g_1 and g_2 have their counts set to zero. Potential group g_{1_2} has the same features and relations as g_1 so it contributes its count, 810, to g_1 and is discarded. Likewise, potential group g_{1_1} matches g_2 so it contributes 90 to g_2 and is discarded. Potential group g_{2_6} also matches g_2, so it contributes 12 to g_2 and is discarded, bringing g_2's total to 102. Potential groups g_{2_1}, g_{2_2}, g_{2_3}, g_{2_4}, and g_{2_5} do not match any extant group, so they become extant groups. The final redistribution of extant groups g_1 and g_2 is:

name	flu	mood	location	count
g_1	s	happy	adams	810.0
g_2	e	annoyed	adams	102.0
g_{2_1}	r	happy	home	12.0
g_{2_2}	r	happy	adams	8.0
g_{2_3}	e	bored	home	30.0
g_{2_4}	e	bored	adams	20.0
g_{2_5}	e	annoyed	home	18.0

By delaying the processing of potential groups until all rules have been applied to all extant groups, PRAM avoids order effects. Imagine that PRAM applied `rule_flu_progression` to g_1 and immediately processed the resulting potential groups, and then applied the rule to g_2. Processing potential group g_{1_1} would make $n_2 = 100 + 90$, and applying the rule to g_2 would redistribute 190 between g_{2_1} and g_{2_2}. Whereas, processing the groups in the opposite order would redistribute 80 between g_{2_1} and g_{2_2}. PRAM eliminates effects of the order of processing of groups. It also eliminates effects of the order of application of rules to groups, as we shall see.

Redistribution Step 3: Iterate. PRAM is designed to explore the dynamics of group counts, so it generally will run iteratively. At the end of each iteration, all non-discarded groups are marked as extant and the preceding steps are repeated: All rules are applied to all extant groups, all potential groups are collected, potential groups that match extant groups are merged with them, and new extant groups are created. A second iteration produces one such new group when the third clause of `rule_flu_progression` is applied to g_{2_1}:

name	flu	mood	location	count
$g_{2\_1\_1}$	s	happy	home	0.24

The reader is invited to calculate the full redistribution resulting from a second iteration (it is surprisingly difficult to do by hand).[1]

6 Exploring Population Dynamics with PRAM

Extending an earlier example, suppose the schools `Adams` and `Berry` each enroll 1000 students, of whom 900 are susceptible and 100 are exposed, evenly divided between males and females. All `Adams` students are middle-income and all `Berry` students are low-income. No students are pregnant, but we add a rule that creates pregnancies in groups of females with probability 0.01. The initial eight extant groups are:

name	flu	sex	income	pregnant	mood	location	count
g1	s	f	m	no	happy	adams	450
g2	e	f	m	no	annoyed	adams	50
g3	s	m	m	no	happy	adams	450
g4	e	m	m	no	annoyed	adams	50
g5	s	f	l	no	happy	berry	450
g6	e	f	l	no	annoyed	berry	50
g7	s	m	l	no	happy	berry	450
g8	e	m	l	no	annoyed	berry	50

The dynamics of `flu=e` at the two schools is presented in Figure 4. The left-most panel shows the proportion of students exposed to flu at each school. `Berry` experiences a strong epidemic, with more than half the students exposed, whereas

[1] The second iteration produces $n_1 = 706.632$, $n_2 = 119.768$, $n_{2\_1} = 26.4$, $n_{2\_1\_1} = 0.24$, $n_{2\_2} = 25.6$, $n_{2\_3} = 60.6$, $n_{2\_4} = 24.4$, $n_{2\_5} = 36.36$.

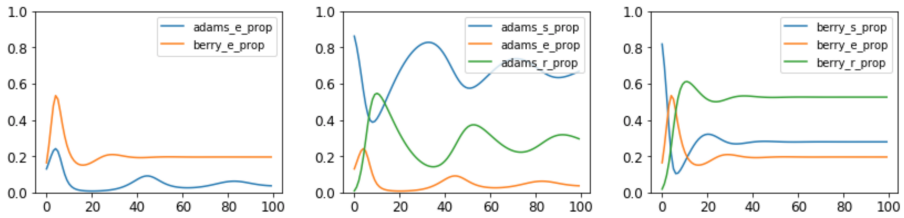

Fig. 4. The left panel shows the proportion of students exposed to flu at `Adams` and `Berry`. The middle and right panels show the proportions of susceptible, exposed and recovered students at each school. The simulation ran for 100 iterations.

`Adams` has a more attenuated epidemic because its students are middle-income and can stay home when they are exposed, thereby reducing the infection probability at the school. `Adams'` endemic level of flu is close to zero whereas `Berry`'s endemic level is around 20%. However, resurgent flu caused by recovered cases becoming susceptible again is more noticeable at `Adams` (around iteration 45). The only difference between `Adams` and `Berry` is that 60% of `Adams` students stay home when they get flu, whereas 10% of `Berry` students do, but this difference has large and persistent consequences.

7 Discussion

PRAM incorporates elements of compartmental models, agent-based models, Markov chain models, dynamic Bayesian models, and probabilistic relational models in a single framework. From compartmental models it takes the idea of homogenous groups (e.g., the group of all individuals exposed to flu), but unlike in compartmental models, PRAM allows for thousands of groups, relationships between groups, and non-stationary probabilities of transitions between groups. Also, PRAM generates groups automatically, whereas compartmental models require the modeler to specify all the compartments at the outset.

By working with groups rather than individuals, PRAM implements lifted inference. The connection between PRAM models and probabilistic models is that counts are proportional to posterior probabilities conditioned on attributes such as `flu` and `has_school` and on the actions of rules that change attributes. PRAM applies rules repeatedly to groups, creating novel groups and merging identical groups, thereby simulating the dynamics of groups' counts. It is in many respects like Probabilistic Relational Models (PRMs) in which groups are defined by their features and relations, but it is designed for simulation and for exploring the dynamics of group counts. There are strong affinities between PRAM models, dynamic Bayesian networks and PRMs, and we are currently working on methods to translate one into another. For example, we have already established model equivalence between PRAM and discrete-time time-homogenous and time-inhomogeneous Markov chains with finite state space. In fact, we have used

these Markov chains to elegantly express some of the well-known epidemiological models: SIS, SIR, and SIRS.

PRAM models may also be viewed as a kind of agent-based model (ABM) in which identical agents constitute groups. This idea offends the tenet of ABMs that agents are unique, but as a practical matter, agents are *not* unique. Consider the roughly two million K12 students in Allegheny County. After mapping age to grade level, mapping nine race classes to four, mapping household size to just three levels, mapping individual households to 350 regions, and ignoring sex, we obtained just 3729 groups. For the purposes of simulating flu dynamics, these mappings are more than generous: Flu affects girls and boys the same, so sex is irrelevant; and race classes might or might not affect flu transmission. Instead of assuming that agents are unique, PRAM models assume that all members of a group are *functionally identical*. Two entities i and j are functionally identical if $\mathcal{F}_i = \mathcal{F}_j$ and $\mathcal{R}_i = \mathcal{R}_j$ after removing all features from $\mathcal{F}_i$ and $\mathcal{F}_j$ and all relations from $\mathcal{R}_i$ and $\mathcal{R}_j$ that are not mentioned in any rule. Said differently, even if two entities have different features and relations, if these don't affect the behavior of a set of rules, then the rules treat these entities in the same way.

The idea of functionally identical groups leads to a population synthesis method. Any feature or relation that is not mentioned in a rule need not be in groups' $\mathcal{F}$ or $\mathcal{R}$, so the only attributes that need to be in groups' definitions are those that condition the actions of rules. Motivated by this observation, we have built a compiler that automatically creates an initial set of groups based on (1) a database that provides $\mathcal{F}$ and $\mathcal{R}$ for individuals and (2) a set of rules. The compiler eliminates from $\mathcal{F}$ and $\mathcal{R}$ those attributes that aren't queried or changed by rules, thereby collapsing a population of individuals into a minimal number of groups. With the help of the compiler, we have been able to construct simulations of an infectious disease spread in the Allegheny County based on a public database of synthetic population of over two million agents.

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 ν 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 ν 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 ν is controlled by the number of groups in the initial population and the actions of rules. Typically, ν grows very quickly to a constant, after which PRAM merely redistributes counts between these groups. In the preceding example, the initial $\nu = 8$ groups grew to $\nu = 44$ on the first iteration and $\nu = 52$ on the second, after which no new groups were added.

By attempting to model processes in various domains we are currently trying to understand the expressive power of PRAM and its limitations. For example, we have implemented a human segregation model which simulates human migration

as a function of similarity and dissimilarity between members of communities [8]. We have also implemented the Poisson point process and subsequently used it to model the incidence of Alzheimer's Disease.

Other research directions we are pursuing include: theoretical work on the equivalence between PRAM models and other model types; investigating the relationship between PRAM and dynamical systems specified via ordinary differential equations (e.g., the Lotka-Volterra population dynamics model); accounting for continuous group features (e.g., income level specified as a probability distribution) and continuous-time simulations (e.g., implementing interacting particle systems); extending the definition of population (e.g., population of values); allowing changes to the total population mass (e.g., birth and death in epidemiological models); ensuring proper amalgamation of different models within the same simulation (e.g., guaranteeing the lack of order effects even when groups from different models interact), and (7) investigating the pedagogical value of PRAM (i.e., deploying it as a teaching tool).

In sum, it appears that several common modeling frameworks are identical with or approximated by redistribution systems and can be implemented in PRAM.

References

1. Blackwood, J.C., Childs, L.M.: An introduction to compartmental modeling for the budding infectious disease modeler. Letters in Biomathematics, 5(1), 195–221 (2018)
2. Getoor, L., Taskar, B. (Eds.) Introduction to statistical relational learning. MIT Press (2007)
3. Grefenstette, J.J., Brown, S.T., Rosenfeld, R., Depasse, J., Stone, N.T., Cooley, P.C., Wheaton, W.D., Fyshe, A., Galloway, D.D., Sriram, A., Guclu, H., Abraham, T., Burke, D.S.: 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, 13(1), 940 (2013)
4. Kermack, W.O., McKendrick, A.G.: A contribution to the mathematical theory of epidemics. Proceedings of the Royal Society A, 115(772), 700–721 (1927)
5. Kravari, K., Bassiliades, N.: A Survey of agent platforms. Journal of Artificial Societies and Social Simulation, 18(1), 11 (2015).
6. Loboda, T.D.: The version of PRAM reported here was developed by the first author. A better engineered version has been developed by the second author and is available at https://github.com/momacs/pram/ (2019)
7. Poole, D.: First-order probabilistic inference. International Joint Conference on Artificial Intelligence (IJCAI-03), 985–991 (2003)
8. Schelling, T.C. Dynamic models of segregation. Journal of Mathematical Sociology, 1, pp. 143–186 (1971).