5-1-2021

Addressing Diversity, Equality, Inclusion and Discrimination By Modeling, Selecting and Ordering Actions

Praneeth Keshav Madabhushi
University of Wisconsin-Milwaukee

ADDRESSING DIVERSITY, EQUALITY, INCLUSION AND DISCRIMINATION BY MODELING, SELECTING AND ORDERING ACTIONS

by

Praneeth Keshav Madabhushi

A Thesis Submitted in

Partial Fulfillment of the

Requirement for the Degree of

Master of Science

in Computer Science

at

The University of Wisconsin-Milwaukee

May 2021

ABSTRACT

ADDRESSING DIVERSITY, EQUALITY, INCLUSION AND DISCRIMINATION BY MODELING, SELECTING AND ORDERING ACTIONS

by

Praneeth Keshav Madabhushi

The University of Wisconsin-Milwaukee. 2021
Under the Supervision of Professor Amol D. Mali

In the current era, Diversity, Equality, and Inclusion (DEI) are often not sufficiently addressed due to bias against certain people or unjust stereotypes or simply an inadequate understanding of the value of DEI. Not addressing DEI sufficiently leads to multiple problems including lawsuits, costly settlements, departure of valuable employees, reduced employee productivity, shortage of qualified workforce, and unjust hiring, compensation, and work-distribution practices. Initiatives to address DEI often fail or risk being ineffective. In this thesis, Food Delivery and Software Company domains are introduced by modeling assignment of drivers to customers who ordered food, and by modeling hiring and project assignment in software industry, using Planning Domain Definition Language (PDDL). This thesis is the first research work to address DEI using fully automated symbolic planning. The experimental results obtained on multiple instances on the 2 domains show that it is possible to express DEI-related objectives using PDDL, and get plans conforming to these objectives. Discrimination is addressed indirectly by nine of the eleven versions of the two domains by enforcing DEI-related objectives. This work can be extended or adapted for use in other domains involving planning or scheduling, to address DEI and discrimination in multiple ways at multiple levels, and the expressive power of PDDL and efficiency of domain-independent as well as domain-specific planners can be exploited in the process.

TABLE OF CONTENTS

LIST OF TABLES

1. INTRODUCTION

Business organizations which do not implement Diversity, Equality, and Inclusion (DEI), are more likely to miss out the innovation and global perspective of employees. Most successful corporate offices in today's world have more prominence of DEI in the workplace. It is important to notice the value of engaging and promoting diverse categories of employees as these employees play a vital role in the reshaping of a company, along with its growth and sustainability.

Diversity is important because if we look from a business perspective, the product which is manufactured by a company would be greatly benefited and refined if the product is offered different perspectives from different backgrounds of employees. For example, in the domain of Online Education, if the teachers who are hired are from different countries and backgrounds, then the material covered in their lecture would be more suitable to everyone as it brings a global perspective in students. Another example is Gaming Industry domain, if the developers are from diverse cultures, then their thoughts and ideas in gaming might reach a large crowd around the globe.

Practicing equality in a company ensures that every employee gets access to equal number of opportunities. If there is no equality, then some employees cannot get along because of overload. Sometimes it may lead to shortage of people if they leave due to work pressure.

If inclusion is not present, then people will feel out of place in environments where they do not see many people who look like them. So, this ends up being an obstacle for many people to enter the industry itself, thereby decreasing the productivity on a large scale. Incorporating inclusion

would make every employee feel at ease and accepted by the company which is an important factor for every organization. In the following chapters, we introduce 2 domains: Food delivery and Software Company and show that it is possible to incorporate DEI by modelling the domains using PDDL and evaluating them using Metric-FF planner.

2. BACKGROUND

2.1 Literature Survey

Addressing DEI issues is not a new trend in current organizations. Efforts have already been made by tech giants. For example, Google has raised $2 million for Black founders to build a more equitable future [1]. According to Rachael Palmer, Head of VC and Startup Partnerships, EMEA “Racial disparity has an adverse effect on everyone and can no longer be ignored. Its damaging effects continue to be reflected in such issues as housing disparities and physical and mental health outcomes, with limited government resources provided to address the root causes. [1]”

Not only in tech companies, but even education domains are also subject to having barriers in the incorporation of DEI. A research study [2] has shown that faculty members have rated male applicants more favorable than female applicants. It is concluded that the faculty participants also selected a higher starting salary and offered more career mentoring to the male applicants when compared to the female applicants selected.

An article on LGBTI [3] and inclusion in private sectors concluded that “Progress will have far-reaching consequences that go beyond the workplace. Attention to employment is part of the larger 2030 Agenda for Sustainable Development, as access to work is deeply intertwined with socio-economic empowerment and the ability to participate in the public sphere. Addressing inequality in the workplace will help to achieve other Sustainable Development Goals including gender equality, decent work and economic growth, and reduced inequalities throughout society.”

A research showed that due to a lack of female STEM students, the Information and Communication Technology sector in Spain is experiencing a skill shortage in the workplace [4]. To these issues, Elizabeth Borneman [5] proposed Digital clinical simulations as a possible solution to the problem of teaching about DEI problems in large-scale educational experiences such as open online courses that are accessible to massive populations.

A survey [6] provided qualitative data on policies aimed at closing the gender gap in the United Kingdom. They provided recommendations like creating programs that encourage female students to pursue engineering at an early age, improving the exposure of women as role models through various social media platforms such as Facebook, Twitter, and technology-related forums.

According to a recent report [7] by American Bar Association, female lawyers reported that they originated more work than some male colleagues but were paid less. The report also documented troubling stories of discrimination that were faced by women working in law firms and claimed thar “Both women who stayed and women who left practice spoke of blatantly unfair compensation systems that are rife with gender bias.” These incidents give a motivation to develop a working prototype that contributes towards DEI incorporation.

2.2 Planning Domain Definition Language and Metric FF

Planning Domain Definition Language (PDDL) is a general-purpose definition language that is useful for designing and modelling a planning or scheduling problem [8]. It was first described for the AIPS-98 planning competition and was later improvised and made efficient to solve more complicated tasks that require expressing time constraints and has time-dependent effects [9].

Now we need a planning system that can find plans based on the instructions given in PDDL. For that purpose, we choose Metric-FF planner. It was developed by Joerg Hoffmann [10] as an extension of standard FF planner [11], that allows numerical state variables. So first we model the problem in PDDL and then run the planner to find a plan that reaches our given goal state.

For modelling a problem, we need two types of files:

1. Domain file: This file describes the definitions of all the actions/operators required to find a plan. We also define all the predicates and functions that are required for each action. For a single domain file, there can be more than one problem file associated with it. So, the domain file for a problem should remain independent of the problem files related to it.
2. Problem file: This file consists of the initial state and goal state along with metric constraints to be reflected in the resulting plan found by the planner. Initial state and goal state are defined using the predicates (true or false variables) that are declared in predicates section of the domain file.

3. NEW SYMBOLIC PLANNING DOMAINS

To address the issues related to DEI, in this Chapter, we introduce two innovative planning models for the following two domains – Food Delivery and Software Company. Each domain is modelled in 5 versions using PDDL. The Software Company domain is also modelled in one extra version.

- Version 1: This is the base version of each domain. This domain file does not consider diversity or equality or inclusion. The planner simply finds a plan to achieve the goal state.
- Version 2: This version of the domain is concerned about diversity. The domain file is designed in such a way that whenever a planner tries to find a plan, it is forced to maximize diversity in the process of achieving the goal state.
- Version 3: This version of the domain is more about ensuring equality in the plan to be found.
- Version 4: This version of the domain ensures diversity and equality in the resulting plan. Extreme cases of version 2 are avoided in this version.
- Version 5: This version of the domain ensures diversity, equality, and inclusion in the resulting plan.
- Version 6 (Software Domain) : This version is used to overshadow the limitations of version 5 in handling edge cases. Description of each version in detailed in presented in the following sections.

3.1 Domain 1: Food Delivery

In this domain, there are drivers for delivering food and customers who can place an order. Drivers can be of different skin-colors (like black drivers and white drivers), similarly customers can also belong to various skin-colors and race. Any driver can deliver food to any customer, with one order at a time. In a single instance, one customer can place only one order, but one driver can deliver for more than one customer in total. Actions related to this domain are about assigning an order of a customer to a driver and delivering an order by a driver. The implementation of these actions varies for each version (mentioned above).

If a food delivery company wants to train their drivers about handing over the order to a customer, then such domain implementation would be of perfect use. If drivers are exposed to delivering and handing over the orders to customers who come from several different backgrounds, then definitely there will be an improvement in quality of service because drivers will know how to approach people in a better way due to diversity exposure in their assigned orders.

3.1.1 Food Delivery: Version 1

This is a basic version of food delivery which assigns orders from customers to drivers using 3 operators. Two operators are about assigning, and one is about delivering.

Both assigning actions assign a customer ‘d’ with driver ‘k’. If the customer-driver pair in the resulting plan is diverse, then it should have been done using Assign-Enforce-diversity action, else using Assign-without-diversity.

Deliver-Order action implicates using predicates that a driver ‘k’ has delivered for customer ‘d’ and makes the driver ‘k’ available for next order.

3.1.2 Food Delivery: Version 2

In this version, diversity in the customer-driver relationship is maximized for the resulting plan. The presence of differences in the entities used in an action would give diversity. In this case, the entities are drivers and customers. For example, if a plan has actions such that black drivers deliver food to white customers and white drivers deliver food to black customers, then such plans promote acceptance of diversity.

Along with deliver-Order action, to ensure maximum diversity, we have 2 operators:

1. Assign-order-without-diversity: This operator is for assigning an order of customer ‘d’ to a driver ‘k’ if the $< d, k >$ pair is not diverse, and there does not exists another driver ‘t’ such that ‘t’ is waiting to be assigned and $< d, t >$ is diverse. This condition makes sure that a non-diverse pair is formed only when there is no diverse pair possible in that state. Now, whether a pair is diverse or not depends on the skin-color of customer and driver. The effects of this operator would set true value for respective predicates like assigned-driver-customer ?d ?k.

2. Assign-Enforce-diversity: This operator assigns an order from customer ‘d’ to driver ‘k’ if the ‘k’ is waiting to be assigned and $< d, k >$ form a diverse pair.

3.1.3 Food Delivery: Version 3

This version makes sure that all orders from customers are assigned in balanced way such that number of orders assigned for each driver is less than the ratio of total number of customers to total number of drivers. To get the count for the number of customers given in the initial state, we introduced an action that initializes the number of customers. Using this count, we can find the ratio, and use this ratio in the preconditions of assignment operators such that any driver ‘t’ who is to be assigned to a customer ‘d’ should have their respective driver load less than the

ratio. This makes ensures equality in the resulting plan by choosing appropriate drivers who satisfy the equality condition for a given action.

3.1.4 Food Delivery: Version 4

This version is a combination of 3.1.2 and 3.1.3. It has two variants of assignment operators as the following:

1. Assign_Enforce_Equality: This action assigns the order of customer ‘d’ to driver ‘k’ if they do not form a diverse pair ,and there does not exist another driver ‘t’ such that $< d, t >$ form a diverse pair and number of orders assigned to driver ‘t’ is lesser than $customer\_count/driver\_count$. This ensures that equality is enforced in the non-diverse pairs of assignments in the plan which used this action.

2. Assign_Enforce_Diversity_Equality: This action assigns the order of customer ‘d’ to driver ‘k’ if they form a diverse pair, and the driver load for ‘k’ is less than $customer\_count/driver\_count$.

3.1.5 Food Delivery: Version 5

A plan promotes inclusion if people with different identities are part of the action. Inclusion is not a byproduct of diversity, in fact, having a diverse plan does not mean that it is inclusive as well. If a plan has actions which makes sure that drivers belonging to minority are also included in the assignment, then that plan is enforced with inclusion. So, in this version, we add a predicate ‘belongs-to-majority’ to achieve this goal. It has the following variants of assigning orders operators:

1. Assign-Enforce-D-E-I: If a plan includes this action, then the customer-driver pair is both diverse and inclusive. This action also makes sure that equality is taken into consideration by comparing driver load with the $customer\_count/driver\_count$. In the effects, diversity count as well as inclusivity count functions are incremented.

2. Assign-Enforce-D-E: If a plan includes this action, then the customer-driver pair is diverse, but not inclusive i.e., driver must belong to majority. Again, equality is enforced during the driver assignment. The effect of the operator increases diversity count.

3. Assign-Enforce-I-E: This action assigns an order from customer ‘d’ to driver ‘k’ when the customer-driver pair is not diverse, but inclusive because driver ‘k’ does not belong to majority, along with maintaining the equality constraint. The effects include increasing the inclusivity count.

4. Assign-Enforce-Equality: This action is used when no action related to diversity and inclusion is not possible to be enforced.

V1	Base version (DEI not enforced for the purpose of comparison with other versions)
V2	Diversity enforced. Requiring the driver’s skin color to be different from the assigned customer’s skin color in as many <customer, driver> pairs as possible.
V3	Equality enforced. Requiring that number of customers assigned to each driver before assigning is less than the ratio of total number of customers to the total number of drivers (upper bound)

V4	DE enforced Requiring the driver's skin color to be different from the assigned customer's skin color in as many <customer, driver> pairs as possible along with maintaining an upper bound for the number of customers assigned to each driver.
V5	DEI enforced Requiring the driver's skin color or race to be different from the assigned customer's skin color or race and driver belonging to minority in as many <customer, driver> pairs as possible along with maintaining an upper bound for the number of customers assigned to each driver.

Table 1 Summary of Food Delivery versions

3.2 Domain 2: Software Company

In this domain, there are applicants who are to be hired for given openings in a software company, and there are projects that are to be assigned to the hired applicants. An applicant may belong to different race, gender, color, etc. For every applicant, the number of years of experience and the highest degree obtained by that applicant is used for hiring. Applicants are considered based on the title for which they applied, and the minimum experience required to be hired for that title. Each applicant can apply only for one title. The number of openings for each title is given in the initial state, and the resulting plan should hire until all openings are filled.

If an applicant is hired, then that applicant can be assigned to various projects. The number of projects along with number of people required for each title of that project is given in the initial state of each problem file.

It is assumed that the number of qualifying applicants for a position is at least as that of the number of openings for that positions, which means that all openings for a title must be filled. Also, if a title requires an experience of ‘n’ years for a candidate with bachelors, then a candidate with master’s degree for the same title would require n-2 years of experience, and a candidate with a PhD would require n-5 years of experience.

3.2.1 Software: Version 1

In this version of Software domain, applicants are hired till all openings for every title is filled using Hiring_Round operator and then assign the hired applicants to projects as required. Every hiring operator hires for an applicant ‘d’ for title ‘t’ only if the number of openings for that title is greater than 0 and the applicants desired title is same as ‘t’. So, to achieve these, we used the following operators:

1. Initialize: Before performing the Hiring action, functions that are related to number of applicants, number of projects and the number of years of experience required for masters and PhD are computed in the initialize operator. We also decide whether an applicant belongs to minority or majority. If an applicant is white, straight, European-American-Non-Hispanic, and not disabled then that applicant comes under majority. All other type of applicants come under minority.

2. Hiring_Round: In this operator, an applicant ‘d’ is hired for title ‘t’ if the number of openings for the title ‘t’ is greater than 0, applicant’s desired title is same as ‘t’, applicant experience is greater than or equals to the minimum experience required for the title and the applicant is not already hired. If these conditions are met, then the applicant is hired and the number of openings for that title is decreased by 1 and the count of number of people hired is increased by 1.

3. Initialize-after-hiring: In this operator, when all openings for each title is filled, then for every applicant who is hired, we initialize the number of projects assigned to that employee to 0.

4. Assign-projects: After initializing-after-hiring, the planner can apply Assign-projects operator to achieve the goal of assigning projects to each of the hired employee till requirements are met. An applicant ‘d’ is assigned to project ‘p’ if the applicant is hired, not previously assigned to the project ‘p’, applicant’s desired title has an opening in that project, i.e., number of required applicants for a project ‘p’ of title ‘t’ is greater than 0 and applicant’s desired title is also ‘t’.

5. Mark-completion-of-assigning: This operator is used to confirm whether all project requirements are met by assigning required number of hired applicants for each title respectively

3.2.2 Software: Version 2

In this version, the hiring of applicants and assigning of projects is done diversely, i.e., the resulting plan after running this domain is such that, from the given group of applicants who are to be hired, every opening for a title is filled such that the applicant who is being hired makes a maximum presence of difference in race, color when compared to the applicants who are already hired for that title. For example, there are applicants D1 who is white, D2 who is white, and D3 who is black, and all of them are qualifying and apply for a title ‘t’ with 2 openings, then hiring D1 and D3 or D2 and D3 would promote maximum diversity when compared to hiring of D1 and D2.

To achieve this, we implement the hiring as well assigning in two rounds. In Round 1 of Hiring, all applicants for a title are hired such that they create maximum presence of difference (like above example), and in Round 2, if there are still titles left to be hired after round 1, then applicants are hired even though they do not make a difference in diversity. Similarly, in Round

1 of Assigning, every title of each project is hired such that it enforces maximum diversity, and the remaining openings of each title for all projects are filled in round 2.

So, to achieve the above in the plan, this version of domain has the following variants of Hiring and Assigning operators:

1. Hiring_Round_1_Diversity: This action hired an applicant ‘d’ for the title ‘t’ if the applicant is qualifying, number of openings for that title is greater than 0 and the applicant should satisfy the diverse condition, i.e., if applicant ‘d’ has to be hired, then there should not exists another applicant ‘c’ such that ‘c’ is hired for the same position and race, skin-color of applicant ‘c’ is same as that of applicant ‘d’. This ensures that in round 1, no two applicants are hired for a title such that they have same race and skin-color. The effects include increasing diversity count and decreasing number of openings for that title.

2. Preparing_for_Hiring_Round_2: This is an action that is used to indicate the planner whether is time for round 2 of hiring or not. So, this an empty action which is only used for checking purpose. The main intention of action is to hold to the planner from using Hiring_Round_2 till the best possible diverse group of applicants is hired for a title in round 1. To do so, we use a predicate named ‘qualified_for_hiring_round_2’ for every qualifying and unhired applicant.

An applicant ‘d’ is qualified for hiring in round 2 only if the applicant is currently not hired and there exists an applicant ‘c’ such that ‘c’ is hired and applicant ‘c’, applicant ‘d’ does not form a diverse pair, i.e., ‘c’ and ‘d’ have same race and skin-color. Now, we can use the predicate ‘qualified_for_hiring_round_2’ for determine if round 2 for a title is necessary or not. For a title to be eligible for round 2, there should not exist an applicant who is not hired for that title and is

not qualified for hiring in round 2. In this way, we can force the planner to use Hiring_for_Round_1 if it possible.

3. Hiring_Round_2: This action is used by the planner to hire an applicant ‘d’ for a title ‘t’ if the applicant is qualified for round 2 and the title ‘t’ requires a round 2 for hiring, along with satisfying the applicant desired title and number of openings for that title constraints.

4. Mark-completion-of-hiring: When this operator is used, it indicates that all required applicants are hired for a title ‘t’ i.e., number of openings for each title should be 0 after hiring.

5. Initialize-after-hiring: This action is invoked after Mark-completion-of-hiring as there is a precondition to this operator that all openings must be filled. The effect in this operator includes initializing the number of projects assigned to each applicant who is hired(employee) to 0.

6. Assign-projects-round-1-diversity: This operator should be used when hiring is completed. So, initializing after hiring is a precondition for this action. This action assigns an application ‘d’ to a project ‘p’ if ‘d’ is hired, number of applicants with title ‘t’ required is greater than 0, applicant is previously not assigned to this project ‘p’, applicant’s desired title is ‘t’ and there does not an applicant ‘c’ such that ‘c’ is already hired and assigned to project ‘p’, ‘c’ and ‘d’ have same race and color. This ensures that applicant ‘d’ is assigned to ‘p’ only if assigning ‘d’ increases diversity within the group of hired applicants who are already assigned to the project ‘p’.

7. Preparing_for_Assigning_Round_2: This action has similar functionality to that of Preparing_for_Hiring_Round_2. But in this case, it checks whether a hired applicant is qualified for round 2 assigning or not. A hired applicant ‘d’ is qualified for round 2 of assigning in a project ‘p’ if there exists another hired applicant ‘c’ such that ‘c’ is assigned to project ‘p’ and ‘c’, ‘d’ pair has same race and color i.e., assigning ‘d’ to project ‘p’ does not increase the

diversity in the group of people assigned to project ‘p’. After checking for each applicant, we can use that information to check whether a project ‘p’ needs round 2 or not for assigning. So, if there does not exist an applicant ‘d’ such that ‘d’ is hired, not assigned to project ‘p’ and does not qualify for round 2 of assigning, then project ‘p’ requires to go for round 2, that is represented using the predicate ‘project_for_round_2’.

8. Assign-projects-round-2: This action is used by the planner to assign projects to hired applicants in round -2 if there are still requirements for applicants in a project ‘p’ for a title ‘t’ after round 1. The fact that this action is used only after round 1 is guaranteed by the predicates ‘qualified_for_round_2_assigning’ and ‘project_for_round_2’ which are computed in Preparing_for_Assigning_Round_2 action.

3.2.3 Software: Version 3

In this version, the focus of implementation is only on enforcing equality in number of projects assigned to each applicant who is hired. So, in this case, the gender, race, skin-color, etc. does not matter. The implementation logic for hiring part remains same as that of version 1, and changes are needed for assigning operator.

We introduce an upper-bound on the number of projects that are to be assigned for a hired applicant. Each hired applicant for title ‘t’ is assigned to a project ‘p’ when that applicant is not previously assigned to ‘p’, and ‘p’ still has a requirement for ‘t’. Along with that, we also check the following upper bound for that applicant: number of projects assigned to an employee should be less than the ratio of total number of required applicants for a title from each project to the total number of applicants hired for that title. This ensures that an employee does not get unfair advantage over others regarding the number of projects assigned.

In the Mark-completion-of-assigning operator, a lower bound condition is used to ensure that each applicant can be assigned with a number of projects such that those number of projects is greater than the ratio of total number of required applicants for a title from each project to the total number of applicants hired for that title minus one. This ensures that every applicant who is hired is assigned with a project whenever total number of required applicants for a title from each project is greater than or equal to number of hired applicants for that title.

3.2.4 Software: Version 4

This version is a combination of version 2 and version 3. The upper bound and lower bound conditions on the number of projects to be assigned for a hired applicant used in version 3 is used in the assignment operators of version 2. This helps in achieving the goal of enforcing diversity along with equality in the domain.

3.2.5 Software: Version 5

In this version, inclusion is incorporated along with diversity and equality. So, an applicant's gender, sexual-orientation and disabilities also matter in this version for hiring and assigning. We take the count of number of people from minority and number of people from majority that are hired for assessing the inclusion metric in this version.

3.2.6 Software: Version 6

One limitation of Version 4 and Version 5 is that in some cases it may return a biased plan. For example, if 25 software engineers need to be hired and there are 25 black applicants and 25 white applicants such that all of them have same race, skin-color, gender, sexual-orientation, and disabilities, then the first round of hiring will hire 1 black applicant and one white applicant as it enforces diversity, but the second round may hire 23 out of the remaining 24 white applicants.

So, the set of hired people includes just 1 black person and 24 white persons, there by creating a bias in the resulting plan.

To address this issue, we introduce Recommended Minimum Acceptance Rates (RMAR) for minority ‘k1’ and majority ‘k2’. The acceptance rates for minority and majority need not be equal, since one will want to hire a higher fraction of minority applicants if there are very few of them and hire a smaller fraction of applicants belonging to majority if there are many of them. But we cannot create goal that is unsatisfiable. We cannot say that more people than needed should be hired or keep openings unfilled.

Now we introduce the following implication that must be checked for title after every opening is filled:

If (k1 * MNA + k2 * MJA) <= (total no. of openings for the title) then,

(k1 * MNA - 1) <= MNH and (k2 * MJA - 1) <= MJH

where, MNA is the no. of minority applicants for a title, MJA is the no. of majority applicants for the same title, MNH is the no. of minority applicants hired for the title and MJH is the no. of majority applicants hired for the title found in the resulting.

The computations of MNA and MJA are done in count_minority_majority_applicants operator whereas the checking of the above constraint is done using an operator named Mark-Completion-of-Checking-acceptance.

V1	Base version (DEI not enforced for the purpose of comparison with other versions)
V2	Diversity enforced Requiring that the applicants are hired for each title and assigned for each project such that presence of difference is maximum. In round 1 of hiring, an applicant is hired for a title if that applicant is qualifying and adds to the diversity among the applicants who are already hired for that title. Remaining applicants are hired in round 2 if required. In round 1 of assigning, an applicant is assigned to a project if that applicant is hired and adds to the diversity among existing applicants in that project. Remaining applicants are assigned to projects in round 2 if required.
V3	Equality enforced Requiring that the number of projects assigned to a hired applicant should be less than the ratio of total number of required applicants for a title from each project to the total number of applicants hired for that title (upper bound). Also, there should not exist an applicant such that the number of projects assigned to an applicant is not lees than the ratio of total number of required applicants for a title from each project to the total number of applicants hired for that title minus one (lower bound).
V4	DE enforced Requiring applicants to be hired and assigned diversely based on race, skin color as much as possible without violating the upper bound and lower bound on the number of projects assigned.
V5	DEI enforced

	Requiring applicants to be hired and assigned diversely based on race, skin color, gender, disabilities, and sexual orientation with inclusion of minority applicants as much as possible without violating the upper bound and lower bound on the number of projects assigned.
V6	DEI enforced with control over hiring rates for majority and minority. Given Recommended Minimum Acceptance Rates (RMAR) for minority ‘k1’ and majority ‘k2’, it requires at least k1 percent of qualified minority applicants and k2 percent of qualified majority applicants to be hired.

Table 2 Summary of Software Company versions

4. EXPERIMENTAL RESULTS

Metric-FF-v2.1 planner was used to output the plans using domain and problem files. It is installed on Ubuntu 20.04.2 LTS 64-bit Operating system, with Intel® Core™ i7-8565U CPU @ 1.80GHz processor, and 12GB RAM.

In every table, DEI metrics should be observed to evaluate the results of each version. The change of these metrics from version to version highlights the robustness and significance of every version. All CPU times in the results are in seconds.

4.1 Food Delivery:

Each domain file is run with 10 problem instances. The insights from each plan are tabulated below:

Name of Problem file	<Customers, Drivers >	Number of actions in the plan found	CPU time needed to find the plan	Diversity Metric
P1	<6,3>	12	0.01	2
P2	<5,2>	10	0.00	2
P3	<10,4>	20	0.00	7
P4	<5,6>	10	0.00	3
P5	<14,7>	28	0.00	10
P6	<16,4>	32	0.00	10
P7	<15,4>	30	0.00	10
P8	<9,5>	18	0.00	6

P9	<6,2>	12	0.00	4
P10	<12,6>	24	0.00	8

Table 3: Food-Delivery Version 1

Name of Problem file	<Customers, Drivers >	Number of actions in the plan found	CPU time needed to find the plan	Diversity Metric
P1	<6,3>	12	0.01	6
P2	<5,2>	10	0.00	5
P3	<10,4>	20	0.00	10
P4	<5,6>	10	0.00	5
P5	<14,7>	28	0.01	14
P6	<16,4>	32	0.00	16
P7	<15,4>	30	0.00	15
P8	<9,5>	18	0.00	9
P9	<6,2>	12	0.00	6
P10	<12,6>	24	0.00	12

Table 4: Food-Delivery Version 2

Name of problem file	<Customers, Drivers>	Number of actions in plan found	CPU time needed	Minimum no. of customers per driver in the plan found	Maximum no. of customers per driver in the plan found
P1	<6,3>	14	0.02	2	2
P2	<5,2>	12	0.00	2	3
P3	<10,4>	22	21.97	2	3
P4	<5,6>	12	0.00	0	1
P5	<14,7>	29	0.00	2	2
P6	<16,4>	33	0.00	4	4
P7	<15,4>	31	0.03	3	4
P8	<9,5>	19	0.00	1	2
P9	<6,2>	13	0.00	3	3
P10	<12,6>	25	0.01	2	2

Table 5: Food-Delivery Version 3

Name of problem file	<Customers, Drivers>	Number of actions in plan found	CPU time needed	Minimum no. of customers per driver	Maximum no. of customers per driver	Diversity metric
P1	<6,3>	13	0.00	2	2	6

P2	<5,2>	11	0.00	2	3	5
P3	<10,4>	21	0.00	1	3	10
P4	<5,6>	11	0.00	0	1	5
P5	<14,7>	29	0.00	2	2	14
P6	<16,4>	33	0.00	4	4	16
P7	<15,4>	32	0.00	3	4	15
P8	<9,5>	18	0.00	1	3	9
P9	<6,2>	12	0.00	2	4	6
P10	<12,6>	24	0.00	1	6	12

Table 6: Food-Delivery Version 4

Name of instance file	<Customers, Drivers>	Number of actions in plan found	CPU time needed	Minimum no. of customers per driver in the plan found	Maximum no. of customers per driver in the plan found	Diversity metric	Inclusivity metric
P1	<6,3>	13	0.00	2	2	5	2
P2	<5,2>	12	0.00	2	3	4	3
P3	<10,4>	21	0.01	1	3	4	9
P4	<5,6>	11	0.01	0	1	4	4
P5	<14,7>	29	0.07	2	2	12	10
P6	<16,4>	33	0.01	4	4	13	12

P7	<15,4>	31	0.00	3	4	13	11
P8	<9,5>	19	0.00	1	2	7	7
P9	<6,2>	13	0.00	3	3	6	3
P10	<12,6>	25	0.00	2	2	11	6

Table 7: Food-Delivery Version 5

4.2 Software Company:

Name of the instance file	No. of minority applicants hired as SE	No. of majority applicants hired as SE	No. of minority applicants hired as Manager	No. of majority applicants hired as Manager
P1	1	1	1	0
P2	2	2	2	0
P3	2	1	0	1
P4	4	0	1	1
P5	5	0	2	0
P6	2	1	1	0
P7	2	1	1	0
P8	3	0	1	1
P9	1	2	1	0
P10	2	2	2	0

Table 8 Software Company Version 1: Part 1

Name of the instance file	No. of minority applicants hired as Senior Manager	No. of majority applicants hired as Senior Manager	No. of actions in the plan found	CPU time needed to find the plan	Diversity metric
P1	n/a	n/a	15	0.00	2
P2	n/a	n/a	17	0.00	4
P3	n/a	n/a	17	0.00	3
P4	n/a	n/a	18	0.00	4
P5	n/a	n/a	19	0.00	4
P6	1	0	16	0.00	4
P7	0	1	18	0.00	3
P8	0	1	21	0.00	3
P9	n/a	n/a	13	0.00	3
P10	n/a	n/a	17	0.00	4

Table 9 Software Company Version 1: Part 2

Name of the instance file	No. of minority applicants hired as SE	No. of majority applicants hired as SE	No. of minority applicants hired as Manager	No. of majority applicants hired as Manager
P1	2	1	1	0

P2	3	1	2	0
P3	2	1	1	0
P4	3	1	0	2
P5	4	1	2	0
P6	3	0	1	0
P7	2	1	1	0
P9	2	1	1	0
P10	3	1	2	0

Table 10 Software Company Version 2: Part 1

Name of the instance file	No. of minority applicants hired as Senior Manager	No. of majority applicants hired as Senior Manager	No. of actions in the plan found	CPU time needed to find the plan	Diversity Metric
P1	n/a	n/a	15	0.63	4
P2	n/a	n/a	17	3.81	6
P3	n/a	n/a	19	0.65	4
P4	n/a	n/a	20	54.58	5
P5	n/a	n/a	21	908.59	5
P6	0	1	16	9.28	5
P7	0	1	18	107.50	5
P9	n/a	n/a	13	0.52	4

P10	n/a	n/a	17	5.53	6

Table 11 Software Company Version 2: Part 2

Name of the instance file	No. of people hired as Software engineers	No. of people hired as managers	No. of people hired as senior managers	Min. no. of projects per software engineer in the plan found	Max. no. of projects per software engineer in the plan found
P1	3	1	n/a	1	2
P2	4	2	n/a	1	2
P3	3	1	n/a	2	2
P4	4	2	n/a	1	2
P5	5	2	n/a	1	2
P6	3	1	1	1	2
P7	3	1	1	2	3
P8	3	2	1	2	3
P9	3	1	n/a	1	2
P10	4	2	n/a	1	2

Table 12 Software Company Version 3: Part 1

Name of the instance file	Min. no. of projects per manager in the plan found	Max. no. of projects per manager in the plan found	Min. no. of projects per senior manager in the plan found	Max. no. of projects per senior manager in the plan found	No. of actions in the plan found	CPU time needed to find the plan
P1	2	2	n/a	n/a	15	0.01
P2	1	1	n/a	n/a	17	0.02
P3	2	2	n/a	n/a	17	0.01
P4	1	1	n/a	n/a	18	0.04
P5	1	1	n/a	n/a	19	0.17
P6	1	1	1	1	16	0.01
P7	1	1	1	1	18	0.21
P8	1	1	2	2	21	1.06
P9	1	1	n/a	n/a	13	0.00
P10	1	1	n/a	n/a	17	0.02

Table 13 Software Company Version 3: Part 2

Name of instance file	No. of minority applicants hired of SE	No. of majority applicants hired as SE	No. of minority applicants hired as Manager	No. of majority applicants hired as Manager	No. of minority applicants hired as Senior Manager	No. of majority applicants hired as Senior Manager
P1	2	1	1	0	n/a	n/a
P2	3	1	2	0	n/a	n/a
P3	2	1	0	1	n/a	n/a
P4	2	1	1	1	n/a	n/a
P5	2	1	1	1	n/a	n/a
P6	2	1	1	0	1	0
P7	2	1	1	0	0	1
P9	2	1	1	0	n/a	n/a
P10	3	1	2	0	n/a	n/a

Table 14 Software Company Version 4: Part 1

Name of instance file	Min. no. of projects per software engineer in the plan found	Max. no. of projects per software engineer in the plan found	Min. no. of projects per manager in the plan found	Max. no. of projects per manager in the plan found	Min. no. of projects per senior manager in the plan found	Max. no. of projects per senior manager in the plan found

P1	1	2	2	2	n/a	n/a
P2	1	2	1	1	n/a	n/a
P3	1	3	2	2	n/a	n/a
P4	1	2	1	1	n/a	n/a
P5	1	2	1	1	n/a	n/a
P6	2	2	1	1	1	1
P7	2	3	1	1	1	1
P9	1	2	1	1	n/a	n/a
P10	1	2	1	1	n/a	n/a

Table 15 Software Company Version 4: Part 2

Name of instance file	No. of actions in the plan found	CPU time needed to find the plan	Diversity count
P1	15	0.95	4
P2	17	4.69	6
P3	19	0.56	4
P4	20	46.66	4
P5	18	805.27	5
P6	16	8.95	5
P7	18	88.04	5
P9	13	0.49	4

P10	17	3.77	6

Table 16 Software Company Version 4: Part 3

Name of instance file	No. of minority applicants hired of SE	No. of majority applicants hired as SE	No. of women, transgender, disabled, or gay people hired as SE	No. of minority applicants hired as Manager	No. of majority applicants hired as Manager	No. of women, transgender, disabled, or gay people hired as Manager
P1	2	1	2	1	0	1
P2	3	1	2	2	0	1
P3	2	1	2	0	1	1
P4	1	2	2	1	1	1
P5	4	1	4	1	1	0
P6	2	1	3	1	0	0
P7	1	2	1	1	0	1
P9	2	1	3	1	0	1
P10	3	1	3	2	0	0

Table 17 Software Company Version 5: Part 1

Name of instance file	No. of minority applicants hired as Senior Manager	No. of majority applicants hired as Senior Manager	No. of women, transgender, disabled, or gay people hired as Senior Manager	Min. no. of projects per SE in the plan found	Max. no. of projects per SE in the plan found
P1	n/a	n/a	n/a	1	2
P2	n/a	n/a	n/a	1	2
P3	n/a	n/a	n/a	2	3
P4	n/a	n/a	n/a	1	2
P5	n/a	n/a	n/a	1	2
P6	1	0	1	1	2
P7	0	1	0	2	3
P9	n/a	n/a	n/a	1	2
P10	n/a	n/a	n/a	1	2

Table 18 Software Company Version 5: Part 2

Name of instance file	Min. no. of projects per manager	Max. no. of projects per manager	Min. no. of projects per senior manager	Max. no. of projects per senior manager	No. of actions in the plan	CPU time needed to find the plan	Diversity count
P1	2	2	n/a	n/a	15	0.74	4

P2	1	1	n/a	n/a	17	3.83	6
P3	2	2	n/a	n/a	17	0.82	4
P4	1	1	n/a	n/a	16	41.09	5
P5	1	1	n/a	n/a	19	841.67	7
P6	1	1	1	1	16	8.31	5
P7	1	1	1	1	18	84.87	5
P9	1	1	n/a	n/a	13	0.54	4
P10	1	1	n/a	n/a	17	7.48	6

Table 19 Software Company Version 5: Part 3

Name of instance file	K1 , K2	No. of minority applicants hired of SE	No. of majority applicants hired as SE	No. of women, transgender, disabled, or gay people hired as SE	No. of minority applicants hired as Manager	No. of majority applicants hired as Manager	No. of women, transgender, disabled, or gay people hired as Manager
P1	0.5, 0.3	1	1	1	1	1	1
P2	0.6, 0.2	3	2	3	n/a	n/a	n/a
P3	0.7, 0.2	1	1	1	0	1	0
P4	0.5, 0.6	0	2	0	0	1	0

Table 20 Software Company Version 6: Part 1

Name of instance file	No. of minority applicants hired as Senior Manager	No. of majority applicants hired as Senior Manager	No. of women, transgender, disabled, or gay people hired as Senior Manager	Min. no. of projects per SE in the plan found	Max. no. of projects per SE in the plan found
P1	n/a	n/a	n/a	1	2
P2	n/a	n/a	n/a	1	2
P3	0	1	0	1	2
P4	0	1	0	1	2

Table 21 Software Company Version 6: Part 2

Name of instance file	Min. no. of projects per manager	Max. no. of projects per manager	Min. no. of projects per senior manager	Max. no. of projects per senior manager	No. of actions in the plan found	CPU time needed to find the plan	Diversity count
P1	1	1	n/a	n/a	15	5.83	4
P2	n/a	n/a	n/a	n/a	17	5.75	5
P3	1	1	1	1	15	6.08	4
P4	1	1	1	1	17	72.59	4

Table 22 Software Company Version 6: Part 3

Observation: It is to be noted that all the applicants who have same attributes are chosen based on the order in which they are defined in the problem. But still, that does not effect the counts of the experiment as it does not matter which applicant it chooses if they have same abilities and properties.

5. FUTURE WORK

A lot of empirical research at the intersection of DEI, discrimination, and planning is possible. Our domains can be modified, or new domains can be developed to include more factors to address DEI and discrimination, including but not limited to political affiliation, age, marital status, immigration status, nationality, religion, height, and specifics of disability. It is possible to address equality by taking into account more factors, including but not limited to details of the work assigned, annual salary, number of promotions, time between promotions, hourly wage, bonus, raises, other benefits including leave allowed, parking options, options for working remotely, locations for working in-person, payment for overtime, and frequency of transfers between physical locations or departments, frequency of job-related travel, screening procedures, disciplinary actions, and flexibility in spreading work hours like 10 hours/day for 4 days/week instead of 8 hours/day for 5 days/week.

Addressing equality thoroughly will also help in preventing discrimination. Addressing DEI and discrimination broadly and deeply is also useful for finding if current versions of PDDL allow a domain designer to represent and verify relevant information in a reasonable time or new versions of PDDL are needed. Addressing DEI and discrimination deeply and broadly using symbolic planning will also help in designing multi-module architectures for addressing DEI and discrimination such that symbolic planning is a module in these architectures such that reasoning, search, other computation, information-gathering, and execution are distributed among the modules for satisfying various criteria including but not limited to efficiency, ease of verification or evaluation of soundness, completeness, optimality, coverage, and cost-effectiveness.

6. CONCLUSION

Most of the past research at the intersection of DEI, discrimination, and AI is on the role of machine learning, automated natural-language processing, and probabilistic reasoning in DEI and discrimination. Automated planning is an extremely important area of AI with an international conference (ICAPS (International Conference on Automated Planning and Scheduling)) dedicated to it. This thesis is the first research work to address DEI using symbolic planning. The experimental results obtained on multiple instances from four of the five versions of Food Delivery domain and five of the six versions of Software Company domain show that it is possible to soundly express DEI-related objectives using PDDL, and get plans conforming to these objectives. Discrimination is addressed indirectly by nine of the eleven versions of the two domains by enforcing DEI-related objectives. Metric FF is a domain-independent planner. Larger instances proved to be challenging to Metric FF.

General lessons learnt that are relevant to other domains are as follows:

(a) DEI can be enforced by carefully defining preconditions and effects of operators

(b) Extra operators may be needed to enforce DEI and the sole purpose of some of the extra operators, preconditions, and effects may be just enforcement of order over other meaningful operators.

(c) Some DEI-related objectives may have to be embedded in operator definitions if the format of goal does not allow them in the goal of the planning instance.

Exploitation of domain-specific knowledge, separation of planning and scheduling components of a domain, and development of representations, preprocessing techniques, and heuristics have helped in speeding up automated plan synthesis by orders of magnitude since 1995. They all have potential to help in solving large instances from complex domains addressing DEI and discrimination very fast. The work in this thesis can be extended or adapted for use in other domains involving planning or scheduling, to address DEI and discrimination in multiple ways at multiple levels, and the expressive power of PDDL and efficiency of domain-independent as well as domain-specific planners can be exploited in the process.

7. REFERENCES

[1] G. f. S. team, Google, 1 Octobor 2020. [Online]. Available: https://blog.google/outreach-initiatives/diversity/2-million-black-founders-equitable-future/. [Accessed 3 May 2021].

[2] "Science faculty's subtle gender biases favor male students," in *Proceedings of the national Academy of Sciences of the United States of America*, 2012.

[3] E. S. Gigi Chao, "The Economist," [Online]. Available: https://prideandprejudice.economist.com/three-ways-private-sector-can-advance-lgbti-rights-inclusion/. [Accessed 2 May 2021].

[4] I. A. Benedé, "Encouraging the Role of Women in the ICT Sector," *IEEE*, Vols. 978-1-7281-0303-7, no. 19, pp. 1832-1835, 2019.

[5] E. Borneman, "Developing Digital Clinical Simulations for Large-Scale," *ACM*, Vols. 978-1-4503-7951-9, no. 20/08, pp. 373-376, 2020.

[6] A. Bennaceur, "Issues in Gender Diversity and Equality in the UK," *ACM/IEEE 1st International Workshop on Gender Equality in Software Engineering*, Vols. 978-1-4503-5738-8, no. 18/05, pp. 5-9, 2018.

[7] L. C. Joyce Sterling, "In their Own Words," American Bar Association, 2021.

[8] M. Ghallab, "PDDL - The Planning Domain Definition Language Version 1.2," Yale Center for Computational Vision and Control, 1998.

[9] D. L. Maria Fox, "Pddl2.1 : An Extension to pddl for Expressing Temporal," *Journal of Artificial Intelligence Research 20 (2003)* , pp. 61-124, 2003.

[10] J. Hoffmann, "Homepage of Metric-FF," [Online]. Available: https://fai.cs.uni-saarland.de/hoffmann/metric-ff.html. [Accessed 17 March 2021].

[11] J. Hoffmann, "Homepage of Fast-Forward," [Online]. Available: https://fai.cs.uni-saarland.de/hoffmann/ff.html. [Accessed 17 March 2021].

8. APPENDICES

Appendix A: Food Delivery Domain

Version 1:

Domain definition:

(define (domain V1_food_delivery)

(:types

driver customer skin-color

)

(:predicates

(waiting-to-be-assigned ?d - driver)

(customer-color ?d - customer ?c1 - skin-color)

(driver-color ?d - driver ?c2 - skin-color)

(orderfor ?d - customer)

(assigned-driver-customer ?x -customer ?y - driver)

(delivered-to ?k - customer)

)

(:action **Assign-Enforce-diversity**

:parameters (?c1 - skin-color ?d -customer ?k - driver)

:precondition (and

(customer-color ?d ?c1)(not(driver-color ?k ?c1))

(orderfor ?d)

(waiting-to-be-assigned ?k)

)

:effect (and

(not(waiting-to-be-assigned ?k))

(not(orderfor ?d))

(assigned-driver-customer ?d ?k)

)

)

(:action **Assign-without-diversity**

:parameters (?c1 - skin-color ?d -customer ?k - driver)

:precondition (and

(customer-color ?d ?c1)

(driver-color ?k ?c1)

(orderfor ?d)

(waiting-to-be-assigned ?k)

)

:effect (and

(not(waiting-to-be-assigned ?k))

(not(orderfor ?d))

(assigned-driver-customer ?d ?k)

)

)

(:action **deliver-order**

:parameters (?d - customer ?k - driver)

:precondition (and (assigned-driver-customer ?d ?k))

:effect (and

(waiting-to-be-assigned ?k)

(not(assigned-driver-customer ?d ?k))

(delivered-to ?d)

)

)

)

Sample problem instance:

(define (problem P6) (:domain V1_food_delivery)

(:objects

c1 c2 c3 c4 c5 c6 c7 c8 c9 c10 c11 c12 c13 c14 c15 c16 - customer

d1 d2 d3 d4 - driver

white black brown - skin-color

)

(:init

(customer-color c1 white) (customer-color c2 white) (customer-color c3 black)

(customer-color c4 brown) (customer-color c5 brown) (customer-color c6 black)

(customer-color c7 brown) (customer-color c8 black) (customer-color c9 black)

(customer-color c10 brown) (customer-color c11 white) (customer-color c12 brown)

(customer-color c13 white) (customer-color c14 white) (customer-color c15 white)

(customer-color c16 black)

(driver-color d1 black) (driver-color d2 brown) (driver-color d3 white)

(driver-color d4 black)

(orderfor c1) (orderfor c2) (orderfor c3) (orderfor c4) (orderfor c5)

(orderfor c6) (orderfor c7) (orderfor c8) (orderfor c9) (orderfor c10)

(orderfor c11) (orderfor c12) (orderfor c13) (orderfor c14) (orderfor c15)

(orderfor c16)

(waiting-to-be-assigned d1) (waiting-to-be-assigned d2) (waiting-to-be-assigned d3)

(waiting-to-be-assigned d4)

)

(:goal (and

(forall (?x -customer)(delivered-to ?x))

))

)

Version 2:

Domain definition:

(define (domain V2_food_delivery)

(:types

driver customer skin-color

)

(:predicates

(waiting-to-be-assigned ?d - driver)

(customer-color ?d - customer ?c - skin-color)

(driver-color ?d - driver ?c - skin-color)

(orderfor ?d - customer)

(assigned-driver-customer ?x -customer ?y - driver)

(delivered-to ?k - customer)

)

(:functions

(diversity_count)

)

(:action **Assign-order-without-diversity**

:parameters (?c - skin-color ?d -customer ?k - driver)

:precondition (and

(orderfor ?d)

(waiting-to-be-assigned ?k)

(customer-color ?d ?c)(driver-color ?k ?c)

(not(exists (?t - driver)(not(driver-color ?t ?c))))

)

:effect (and

(not(waiting-to-be-assigned ?k))

(not(orderfor ?d))

(assigned-driver-customer ?d ?k)

)

)

(:action **Assign-Enforce-diversity**

:parameters (?c - skin-color ?d -customer ?k - driver)

:precondition (and

(orderfor ?d)

(waiting-to-be-assigned ?k)

(customer-color ?d ?c)(not(driver-color ?k ?c))

)

:effect (and

(not (waiting-to-be-assigned ?k))

(not(orderfor ?d))

(increase (diversity_count) 1)

(assigned-driver-customer ?d ?k)

)

)

(:action **deliver-order**

:parameters (?d - customer ?k - driver)

:precondition (and (assigned-driver-customer ?d ?k))

:effect (and

(waiting-to-be-assigned ?k)

(not(assigned-driver-customer ?d ?k))

(delivered-to ?d)

))

)

Sample problem instance:

(define (problem P6) (:domain diverse_food_delivery)

(:objects

c1 c2 c3 c4 c5 c6 c7 c8 c9 c10 c11 c12 c13 c14 c15 c16 - customer

d1 d2 d3 d4 - driver

white black brown - skin-color

)

(:init

(=(diversity_count) 0)

(customer-color c1 white)(customer-color c2 white)(customer-color c3 black)

(customer-color c4 brown)(customer-color c5 brown)(customer-color c6 black)

(customer-color c7 brown)(customer-color c8 black)(customer-color c9 black)

(customer-color c10 brown)(customer-color c11 white)(customer-color c12 brown)

(customer-color c13 white)(customer-color c14 white)(customer-color c15 white)

(customer-color c16 black)(driver-color d1 black)(driver-color d2 brown)

(driver-color d3 white)(driver-color d4 black)(orderfor c1)

(orderfor c2)(orderfor c3)(orderfor c4)(orderfor c5)(orderfor c6)

(orderfor c7)(orderfor c8)(orderfor c9)(orderfor c10)(orderfor c12)

(orderfor c13)(orderfor c14)(orderfor c15)(orderfor c16)

(waiting-to-be-assigned d1)(waiting-to-be-assigned d2)

(waiting-to-be-assigned d3)(waiting-to-be-assigned d4)

)

(:goal (and

(forall (?x -customer)(delivered-to ?x))

)))

Version 3:

Domain definition:

(define (domain V3_food_delivery)

(:types

driver customer

)

(:predicates

(orderfor ?d - customer)

(waiting-to-be-assigned ?t - driver)

(assigned-driver-customer ?d - customer ?t - driver)

(delivered-to ?d - customer)

(customers_counted)

)

```
(:functions
(driver_load ?t - driver)
(customer_driver_ratio)
(customer_count)
(driver_count)
)
(:action count_customers
:parameters ()
:precondition (and )
:effect (and
(forall (?d - customer) (increase (customer_count) 1))
(forall (?t - driver) (assign (driver_load ?t) 0))
(customers_counted)
)
)
(:action Assign-Enforce-equality
:parameters (?d - customer ?t - driver)
```

```
:precondition (and
(customers_counted)
(orderfor ?d)
(waiting-to-be-assigned ?t)
(<(driver_load ?t) (/ (customer_count) (driver_count)))
)
:effect (and
(assigned-driver-customer ?d ?t)
(not (waiting-to-be-assigned ?t))
(not (orderfor ?d))
(increase (driver_load ?t) 1)
)
)
(:action deliver-order
:parameters (?d - customer ?t - driver)
:precondition (and
(assigned-driver-customer ?d ?t)
```

)

:effect (and

(delivered-to ?d)

(waiting-to-be-assigned ?t)

))

)

Note: It is possible to count the number of drivers from the initial state for computing customer-driver ratio, but the count for number of drivers was given manually in the initial state because Metric-FF planner does not support non-linear tasks on the functions whose values have been changed due to increment or decrement operators used on it i.e., suppose if we initialize a function named driver_count to 0 in initial state, and in the domain file we increment it by 1 for every driver to get the count, then we cannot perform customer_count/driver_count because drivers_count has been incremented and used in the denominator of the ratio, whereas, if we initialized it in the initial state directly with the number of drivers, then the computation of the ratio is supported in metric-ff.

Sample problem instance:

(define (problem P6) (:domain V3_food_delivery)

(:objects

c1 c2 c3 c4 c5 c6 c7 c8 c9 c10 c11 c12 c13 c14 c15 c16 - customer

d1 d2 d3 d4 - driver

)

(:init

(orderfor c1)(orderfor c2)(orderfor c3)

(orderfor c4)(orderfor c5)(orderfor c6)

(orderfor c7)(orderfor c8)(orderfor c9)

(orderfor c10)(orderfor c11)(orderfor c12)

(orderfor c13)(orderfor c14)(orderfor c15)

(orderfor c16)

(waiting-to-be-assigned d1)

(waiting-to-be-assigned d2)

(waiting-to-be-assigned d3)

(waiting-to-be-assigned d4)

(=(customer_count) 0)

(=(driver_count) 4)

)

(:goal (and

(forall (?x -customer)(delivered-to ?x))

))

)

Version 4:

Domain definition:

(define (domain V4_food_delivery)

(:types

driver customer skin-color

)

(:predicates

(waiting-to-be-assigned ?d - driver)

(customer-color ?d - customer ?c - skin-color)

(driver-color ?d - driver ?c - skin-color)

(orderfor ?d - customer)

(initialized)

(assigned-driver-customer ?x -customer ?y - driver)

(delivered-to ?k - customer)

)

(:functions

(diversity_count)

(driver_count)

(customer_count)

(driver_load ?d - driver)

)

(:action **initialize-driver-loads**

:parameters ()

:precondition (and

)

:effect (and

(forall (?d - driver) (assign (driver_load ?d) 0))

(forall (?c - customer) (increase (customer_count) 1))

(initialized)

)

)

(:action **Assign_Enforce_Equality**

:parameters (?c - skin-color ?d -customer ?k - driver)

:precondition (and

(initialized)

(orderfor ?d)

(waiting-to-be-assigned ?k)

(customer-color ?d ?c)(driver-color ?k ?c)

(not(exists (?t - driver)(and(not(driver-color ?t ?c))(<(driver_load ?t)

(/(customer_count)(driver_count))))))

(<(driver_load ?k) (/(customer_count)(driver_count)))

)

:effect (and

(not(waiting-to-be-assigned ?k))

(not(orderfor ?d))

(increase (driver_load ?k) 1)

(assigned-driver-customer ?d ?k)

)

)

(:action **Assign_Enforce_Diversity_Equality**

:parameters (?c - skin-color ?d -customer ?k - driver)

:precondition (and

(initialized)

(orderfor ?d)

(waiting-to-be-assigned ?k)

(customer-color ?d ?c)(not(driver-color ?k ?c))

(<(driver_load ?k) (/(customer_count)(driver_count)))

)

:effect (and

(not (waiting-to-be-assigned ?k))

(not(orderfor ?d))

(increase (diversity_count) 1)

(increase (driver_load ?k) 1)

(assigned-driver-customer ?d ?k)

)

)

(:action **deliver-order**

:parameters (?d - customer ?k - driver)

:precondition (and (assigned-driver-customer ?d ?k))

:effect (and

(waiting-to-be-assigned ?k)

(not(assigned-driver-customer ?d ?k))

(delivered-to ?d)

))

)

Sample problem instance:

(define (problem P6) (:domain V4_food_delivery)

(:objects

c1 c2 c3 c4 c5 c6 c7 c8 c9 c10 c11 c12 c13 c14 c15 c16 - customer

d1 d2 d3 d4 - driver

white black brown - skin-color

)

(:init

(=(diversity_count) 0)(=(customer_count) 0)(=(driver_count) 4)

(customer-color c1 white)(customer-color c2 white)

(customer-color c3 black)(customer-color c4 brown)

(customer-color c5 brown)(customer-color c6 black)

(customer-color c7 brown)(customer-color c8 black)

(customer-color c9 black)(customer-color c10 brown)

(customer-color c11 white)(customer-color c12 brown)

(customer-color c13 white)(customer-color c14 white)

(customer-color c15 white)(customer-color c16 black)

(driver-color d1 black)(driver-color d2 brown)

(driver-color d3 white)(driver-color d4 black)

(orderfor c1)(orderfor c2)(orderfor c3)(orderfor c4)

(orderfor c5)(orderfor c6)(orderfor c7)(orderfor c8)

(orderfor c9)(orderfor c10)(orderfor c11)(orderfor c12)

(orderfor c13)(orderfor c14)(orderfor c15)(orderfor c16)

(waiting-to-be-assigned d1)(waiting-to-be-assigned d2)

(waiting-to-be-assigned d3)(waiting-to-be-assigned d4)

)

(:goal (and

(forall (?x -customer)(delivered-to ?x))

)))

Version 5:

Domain definition:

(define (domain V5_food_delivery)

(:types

driver customer skin-color race)

(:predicates

(waiting-to-be-assigned ?d - driver)

(customer-color-race ?d - customer ?c1 - skin-color ?r1 - race)

(driver-color-race ?d - driver ?c2 - skin-color ?r2 - race)

(orderfor ?d - customer)

(initialized)

(assigned-driver-customer ?x -customer ?y - driver)

(delivered-to ?k - customer)

(belongs-to-majority ?d - driver)

)

```
(:functions
(diversity_count)
(inclusivity_count)
(driver_count)
(customer_count)
(driver_load ?d - driver)
)
(:action initialize-driver-loads
:parameters ()
:precondition (and
)
:effect (and
(forall (?d - driver) (assign (driver_load ?d) 0))
(forall (?c - customer) (increase (customer_count) 1))
(initialized)
)
)
```

(:action **Assign-Enforce-D-E-I**

:parameters (?c1 - skin-color ?r1 - race ?d -customer ?k - driver)

:precondition (and

(initialized)

(orderfor ?d)

(waiting-to-be-assigned ?k)

(not(belongs-to-majority ?k))

(customer-color-race ?d ?c1 ?r1)(not(driver-color-race ?k ?c1 ?r1))

(<(driver_load ?k) (/(customer_count)(driver_count)))

)

:effect (and

(not(waiting-to-be-assigned ?k))

(not(orderfor ?d))

(increase (driver_load ?k) 1)

(assigned-driver-customer ?d ?k)

(increase (diversity_count) 1)

(increase (inclusivity_count) 1)

)

)

(:action **Assign-Enforce-D-E**

:parameters (?c1 - skin-color ?r1 - race ?d -customer ?k - driver)

:precondition (and

(initialized)

(orderfor ?d)

(belongs-to-majority ?k)

(waiting-to-be-assigned ?k)

(customer-color-race ?d ?c1 ?r1)(not(driver-color-race ?k ?c1 ?r1))

(<(driver_load ?k) (/(customer_count)(driver_count)))

)

:effect (and

(not(waiting-to-be-assigned ?k))

(not(orderfor ?d))

(increase (driver_load ?k) 1)

(increase (diversity_count) 1)

(assigned-driver-customer ?d ?k)

)

)

(:action **Assign-Enforce-I-E**

:parameters (?c1 - skin-color ?r1 - race ?d -customer ?k - driver)

:precondition (and

(initialized)

(orderfor ?d)

(not(belongs-to-majority ?k))

(waiting-to-be-assigned ?k)

(customer-color-race ?d ?c1 ?r1)(driver-color-race ?k ?c1 ?r1)

(<(driver_load ?k) (/(customer_count)(driver_count)))

)

:effect (and

(not(waiting-to-be-assigned ?k))

(not(orderfor ?d))

(increase (driver_load ?k) 1)

(increase (inclusivity_count) 1)

(assigned-driver-customer ?d ?k)

)

)

(:action **Assign-Enforce-Equality**

:parameters (?c1 - skin-color ?r1 - race ?d -customer ?k - driver)

:precondition (and

(initialized)

(orderfor ?d)

(waiting-to-be-assigned ?k)

(customer-color-race ?d ?c1 ?r1)

(<(driver_load ?k) (/(customer_count)(driver_count)))

(not(exists (?t - driver)(and(or (not(driver-color-race ?t ?c1 ?r1))(not(belongs-to-majority ?t)))

(<(driver_load ?t) (/(customer_count)(driver_count))))))

)

:effect (and

(not(waiting-to-be-assigned ?k))

(not(orderfor ?d))

(increase (driver_load ?k) 1)

(assigned-driver-customer ?d ?k)

)

)

(:action **deliver-order**

:parameters (?d - customer ?k - driver)

:precondition (and (assigned-driver-customer ?d ?k))

:effect (and

(waiting-to-be-assigned ?k)

(not(assigned-driver-customer ?d ?k))

(delivered-to ?d)

))

)

Sample problem instance:

(define (problem P6) (:domain V5_food_delivery)

(:objects

c1 c2 c3 c4 c5 c6 c7 c8 c9 c10 c11 c12 c13 c14 c15 c16 - customer

d1 d2 d3 d4 - driver

white black brown - skin-color

Asian African-American European-American-Non-Hispanic American-Indian - race

)

(:init

(=(diversity_count) 0)(=(customer_count) 0)

(=(driver_count) 4)(=(inclusivity_count) 0)

(customer-color-race c1 white European-American-Non-Hispanic)

(customer-color-race c2 white European-American-Non-Hispanic)

(customer-color-race c3 black African-American)

(customer-color-race c4 brown American-Indian)

(customer-color-race c5 brown American-Indian)

(customer-color-race c6 black African-American)

(customer-color-race c7 brown American-Indian)

(customer-color-race c8 black African-American)

(customer-color-race c9 black African-American)

(customer-color-race c10 brown American-Indian)

(customer-color-race c11 white European-American-Non-Hispanic)

(customer-color-race c12 brown American-Indian)

(customer-color-race c13 white European-American-Non-Hispanic)

(customer-color-race c14 white European-American-Non-Hispanic)

(customer-color-race c15 white European-American-Non-Hispanic)

(customer-color-race c16 black African-American)

(driver-color-race d1 black African-American)

(driver-color-race d2 brown American-Indian)

(driver-color-race d3 white European-American-Non-Hispanic)

(driver-color-race d4 black African-American)

(orderfor c1)(orderfor c2)(orderfor c3)

(orderfor c4)(orderfor c5)(orderfor c6)

(orderfor c7)(orderfor c8)(orderfor c9)

(orderfor c10)(orderfor c11)(orderfor c12)

(orderfor c13)(orderfor c14)(orderfor c15)

(orderfor c16)

(waiting-to-be-assigned d1)(waiting-to-be-assigned d2)

(waiting-to-be-assigned d3)(waiting-to-be-assigned d4)

(belongs-to-majority d3)

)

(:goal (and

(forall (?x -customer)(delivered-to ?x))

))

)

Appendix B: Software Company

Version 1:

Domain definition:

(define (domain V1_Software)

(:types

applicant title project

gender race sexual-orientation

disabilities color

)

(:predicates

(IsHired ?d - applicant)

(applicant_for_title ?d - applicant ?t - title)

(hasBachelors ?d - applicant)

(hasMasters ?d - applicant)

(hasPhD ?d - applicant)

(initialized)

assigned_applicant_to_project ?d - applicant ?p - project)

(hiring_filled_all_titles)

(employee_project_counts_initialized)

(all-assigned)

(isWhite ?d - applicant)

(isTransgender ?d - applicant)

(isEuropean-American-Non-Hispanic ?d - applicant)

(isStraight ?d - applicant)

(isDisabled ?d - applicant)

(belongs-to-majority ?d - applicant)

(belongs-to-minority ?d - applicant)

(applicant-race ?d - applicant ?r - race)

(applicant-gender ?d - applicant ?g - gender)

(applicant-color ?d - applicant ?c - color)

(applicant-so ?d - applicant ?so - sexual-orientation)

(applicant-disabilities ?d - applicant ?ds - disabilities)

(isNotstraight ?d - applicant)

)

(:functions

(applicants_count)

(applicant_experience ?d - applicant)

(minimum_experience_required_for_title ?t - title)

(number_of_openings_per_title ?t - title)

(projects_count)

(titles_per_project ?t - title ?p - project) ; how many SE, Managers per project

(number_of_projects_assigned_to_employee ?d - applicant)

(Hired_count)

(number_of_titles_for_hiring ?t - title)

(minority_applicant_count)

(majority_applicant_count)

(minority_applicants_hired)

(majority_applicants_hired)

)

;define actions here

;Masters should apply for titles with experience_required > 2 and PHd with > 5

(:action **Initialize**

```
:parameters ()

:precondition (and

(not(initialized))

)

:effect (and

(forall (?c - applicant) (and(increase (applicants_count) 1)))

(forall (?c - applicant) (when (hasMasters ?c) (increase (applicant_experience ?c) 2) ))

(forall (?c - applicant) (when (hasPhD ?c) (increase (applicant_experience ?c) 5) ))

(forall (?p - project) (and (increase (projects_count) 1)))

(forall (?d - applicant) (when (and (isWhite ?d)(not(isNotStraight ?d))(not(isTransgender ?d))(isEuropean-American-Non-Hispanic ?d)(not (isDisabled ?d)))

(and (belongs-to-majority ?d)(increase (majority_applicant_count) 1))

)

)

(forall (?d - applicant) (when (or(isNotStraight ?d)(isDisabled ?d)(isTransgender ?d)(not(and(isWhite ?d)(isEuropean-American-Non-Hispanic ?d)))) ; Asians can be white

(and (belongs-to-minority ?d)(increase (minority_applicant_count) 1))    )

)
```

(initialized)

)

)

(:action **Hiring_Round**

:parameters (?d - applicant ?t - title)

:precondition (and

(initialized)

(>(number_of_openings_per_title ?t) 0)

(applicant_for_title ?d ?t)

(>=(applicant_experience ?d)(minimum_experience_required_for_title ?t))

(not(IsHired ?d))

)

:effect (and

(IsHired ?d)

(decrease (number_of_openings_per_title ?t) 1)

(increase (Hired_count) 1)

(when (belongs-to-minority ?d) (increase (minority_applicants_hired) 1))

(when (belongs-to-majority ?d) (increase (majority_applicants_hired) 1))

)

)

(:action **Mark-completion-of-hiring**

:parameters ( )

:precondition (and

(initialized)

(not(hiring_filled_all_titles))

)

:effect (and

(when (forall (?t - title) (and (=(number_of_openings_per_title ?t) 0))) (hiring_filled_all_titles)))

)

(:action **Initialize-after-hiring**

:parameters ( )

:precondition (and

(hiring_filled_all_titles)

(not(employee_project_counts_initialized))

)

:effect (and

(forall (?d - applicant) (when(isHired ?d) (assign (number_of_projects_assigned_to_employee ?d) 0))

) (employee_project_counts_initialized))

)

(:action **Assign-projects**

:parameters (?d - applicant ?t - title ?p - project)

:precondition (and

(employee_project_counts_initialized)

(isHired ?d)

(not(assigned_applicant_to_project ?d ?p))

(applicant_for_title ?d ?t)

(>(titles_per_project ?t ?p) 0)

)

:effect (and

(assigned_applicant_to_project ?d ?p)

(increase(number_of_projects_assigned_to_employee ?d) 1)

(decrease(titles_per_project ?t ?p) 1)

))

(:action **Mark-completion-of-assigning**

:parameters ()

:precondition (and

(not(all-assigned))

)

:effect (and

;when every project is assigned with required count of titles

(when (forall(?t - title ?p - project)(and(=(titles_per_project ?t ?p)0))) (all-assigned)))

)

)

Sample problem instance:

(define (problem P8) (:domain V1_software)

(:objects

a1 a2 a3 a4 a5 a6 a7 a8 a9 a10 a11 - applicant

SE Manager Senior-Manager - title

p1 p2 p3 p4 - project

Asian European-American-Non-Hispanic African-American American-Indian - race

white black brown - color

male female transgender - gender

straight gay - sexual-orientation

disabled none - disabilities

)

(:init

(applicant-race a1 American-Indian)(applicant-so a1 straight)

(applicant-gender a1 female)(applicant-disabilities a1 none)

(applicant-color a1 brown)

(applicant-race a2 African-American)(applicant-so a2 straight)

(applicant-gender a2 male)(applicant-disabilities a2 none)

(applicant-color a2 black)

(applicant-race a3 African-American)(applicant-so a3 gay)

(applicant-gender a3 male)(applicant-disabilities a3 none)

(applicant-color a3 black)

(applicant-race a4 Asian)(applicant-so a4 straight)

(applicant-gender a4 male)(applicant-disabilities a4 none)

(applicant-color a4 brown)

(applicant-race a5 European-American-Non-Hispanic)(applicant-so a5 straight)

(applicant-gender a5 female)(applicant-disabilities a5 none)

(applicant-color a5 white)(applicant-race a6 African-American)

(applicant-so a6 straight)(applicant-gender a6 female)

(applicant-disabilities a6 disabled)(applicant-color a6 black)

(applicant-race a7 European-American-Non-Hispanic)(applicant-so a7 straight)

(applicant-gender a7 female)(applicant-disabilities a7 none)

(applicant-color a7 white)(applicant-race a8 European-American-Non-Hispanic)

(applicant-so a8 straight)(applicant-gender a8 male)

(applicant-disabilities a8 none)(applicant-color a8 white)

(applicant-race a9 African-American)(applicant-so a9 straight)

(applicant-gender a9 transgender)(applicant-disabilities a9 none)

(applicant-color a9 black)(applicant-race a10 European-American-Non-Hispanic)

(applicant-so a10 straight)(applicant-gender a10 female)

(applicant-disabilities a10 none)(applicant-color a10 white)

(applicant-race a11 European-American-Non-Hispanic)

(applicant-so a11 straight)(applicant-gender a11 male)

(applicant-disabilities a11 none)(applicant-color a11 white)

(isWhite a5)(isWhite a7)(isWhite a8)(isWhite a10)(isWhite a11)

(isEuropean-American-Non-Hispanic a5)

(isEuropean-American-Non-Hispanic a7)(isEuropean-American-Non-Hispanic a8)(isEuropean-American-Non-Hispanic a10)

(isEuropean-American-Non-Hispanic a11)

(isDisabled a6)(isNotstraight a3)(isTransgender a9)

(=(applicants_count) 0)(=(projects_count) 0)

(=(minority_applicant_count) 0)(=(majority_applicant_count)0)

(=(minority_applicants_hired)0)(=(majority_applicants_hired)0)

(=(Hired_count)0)

(=(titles_per_project SE p1) 2)

(=(titles_per_project SE p2) 3)

(=(titles_per_project SE p3) 0)

(=(titles_per_project SE p4) 2)

(=(titles_per_project Manager p1) 0)

(=(titles_per_project Manager p2) 1)

(=(titles_per_project Manager p3) 1)

(=(titles_per_project Manager p4) 0)

(=(titles_per_project Senior-Manager p1) 1)

(=(titles_per_project Senior-Manager p2) 0)

(=(titles_per_project Senior-Manager p3) 0)

(=(titles_per_project Senior-Manager p4) 1)

(=(number_of_openings_per_title SE) 3)

(=(number_of_openings_per_title Manager) 2)

(=(number_of_openings_per_title Senior-Manager) 1)

(=(minimum_experience_required_for_title SE) 3)

(=(minimum_experience_required_for_title Manager) 7)

(=(minimum_experience_required_for_title Senior-Manager) 11)

(=(number_of_titles_for_hiring SE) 3)

(=(number_of_titles_for_hiring Manager) 2)

(=(number_of_titles_for_hiring Senior-Manager) 1)

(=(applicant_experience a1) 2)(=(applicant_experience a2) 4)

(=(applicant_experience a3) 3)(=(applicant_experience a4) 3)

(=(applicant_experience a5) 10)(=(applicant_experience a6) 5)

(=(applicant_experience a7) 3)(=(applicant_experience a8) 9)

(=(applicant_experience a9) 7)(=(applicant_experience a10) 3)

(=(applicant_experience a11) 8)

(applicant_for_title a1 SE)(hasMasters a1)

(applicant_for_title a2 SE)(hasBachelors a2)

(applicant_for_title a3 SE)(hasMasters a3)

(applicant_for_title a4 Manager)(hasMasters a4)

(applicant_for_title a5 Senior-Manager)(hasPhD a5)

(applicant_for_title a6 SE)(hasBachelors a6)

(applicant_for_title a7 Manager)(hasBachelors a7)

(applicant_for_title a8 Senior-Manager)(hasPhD a8)

(applicant_for_title a9 Manager)(hasPhD a9)

(applicant_for_title a10 SE)(hasMasters a10)

(applicant_for_title a11 Manager)(hasMasters a11)

)

(:goal (and (all-assigned))))

Version 2:

Domain definition:

(define (domain V2_Software)

(:requirements :typing :fluents :negative-preconditions :conditional-effects :disjunctive-preconditions)

(:types

applicant title project

gender race sexual-orientation

disabilities color

)

(:predicates

(initialized)(all-assigned)

(hiring_filled_all_titles)(employee_project_counts_initialized)

(IsHired ?d - applicant)(applicant_for_title ?d - applicant ?t - title)

(hasBachelors ?d - applicant)(hasMasters ?d - applicant)

(hasPhD ?d - applicant)(isWhite ?d - applicant)

(isTransgender ?d - applicant)(isEuropean-American-Non-Hispanic ?d - applicant)

(isNotStraight ?d - applicant)(isDisabled ?d - applicant)

(isNotstraight ?d - applicant)

(applicant-race ?d - applicant ?r - race)

(applicant-gender ?d - applicant ?g - gender)

(applicant-color ?d - applicant ?c - color)

(applicant-so ?d - applicant ?so - sexual-orientation)

(applicant-disabilities ?d - applicant ?ds - disabilities)

(qualified_for_round_2_assigning ?d - applicant ?p - project)

(project_for_round_2 ?p - project)(qualified_for_hiring_round_2 ?d - applicant)

(can_hire_in_round_2 ?t - title)

(belongs-to-majority ?d - applicant)

(belongs-to-minority ?d - applicant)

(assigned_employee_to_project ?d - applicant ?p - project)

)

(:functions

(applicants_count)

(applicant_experience ?d - applicant)

(minimum_experience_required_for_title ?t - title)

(number_of_openings_per_title ?t - title)

(projects_count)

(titles_per_project ?t - title ?p - project) ; how many SE, Managers per project

(number_of_projects_assigned_to_employee ?d - applicant)

(Hired_count)

(number_of_titles_for_hiring ?t - title)

(minority_applicant_count)

(majority_applicant_count)

(minority_applicants_hired)

(majority_applicants_hired)

(projects_diversity_count)

(hired_for_diversity_count)

)

(:action **Initialize**

:parameters ()

```
:precondition (and
(not(initialized))
)
:effect (and
(forall (?c - applicant) (and(increase (applicants_count) 1)))
(forall (?c - applicant) (when (hasMasters ?c) (increase (applicant_experience ?c) 2) ))
(forall (?c - applicant) (when (hasPhD ?c) (increase (applicant_experience ?c) 5) ))
(forall (?p - project) (and (increase (projects_count) 1)))
(forall (?d - applicant) (when (and (isWhite ?d)(not(isNotStraight ?d))(not(isTransgender
?d))(isEuropean-American-Non-Hispanic ?d)(not (isDisabled ?d)))
(and (belongs-to-majority ?d)(increase (majority_applicant_count) 1))
) )
(forall (?d - applicant) (when (or(isNotStraight ?d)(isDisabled ?d)(isTransgender
?d)(not(and(isWhite ?d)(isEuropean-American-Non-Hispanic ?d)))) ; Asians can be white
(and (belongs-to-minority ?d)(increase (minority_applicant_count) 1))
) )
(initialized) )
)
```

(:action **Hiring_Round_1_Diversity**

:parameters (?d - applicant ?t - title)

:precondition (and

(initialized)

(>(number_of_openings_per_title ?t) 0)

(applicant_for_title ?d ?t)

(>=(applicant_experience ?d)(minimum_experience_required_for_title ?t))

(not(IsHired ?d))

(not(exists (?c - applicant)

(and

(IsHired ?c)

(applicant_for_title ?c ?t)

(>=(applicant_experience ?c)(minimum_experience_required_for_title ?t))

(exists (?r - race)(and(applicant-race ?c ?r)(applicant-race ?d ?r)))

(exists (?k - color)(and(applicant-color ?c ?k)(applicant-color ?d ?k)))

))

))

:effect (and

(IsHired ?d)

(decrease (number_of_openings_per_title ?t) 1)

(increase (Hired_count) 1)

(increase (hired_for_diversity_count) 1)

(when (belongs-to-minority ?d) (increase (minority_applicants_hired) 1))

(when (belongs-to-majority ?d) (increase (majority_applicants_hired) 1))

)

)

(:action **Preparing_for_Hiring_Round_2**

:parameters ()

:precondition (and

(initialized)

)

:effect (and

(forall (?d - applicant)

(when (and

```
(not (IsHired ?d))

(not(qualified_for_hiring_round_2 ?d))

(exists (?c - applicant)

(and

(IsHired ?c)

(or

(exists (?r - race)(and(applicant-race ?c ?r)(applicant-race ?d ?r)))

(exists (?k - color)(and(applicant-color ?c ?k)(applicant-color ?d ?k)))

) ) )

)

(and

(qualified_for_hiring_round_2 ?d)

) ) )

(forall (?t - title)

(when ( and

(not (exists (?d - applicant)

(and
```

(applicant_for_title ?d ?t)

(not(IsHired ?d))

(not(qualified_for_hiring_round_2 ?d))

)))

)

(and

(can_hire_in_round_2 ?t)

)))

(:action **Hiring_Round_2**

:parameters (?d - applicant ?t - title)

:precondition (and

(qualified_for_hiring_round_2 ?d)

(>(number_of_openings_per_title ?t) 0)

(applicant_for_title ?d ?t)

(can_hire_in_round_2 ?t)

(>=(applicant_experience ?d)(minimum_experience_required_for_title ?t))

(not(IsHired ?d))

)

:effect (and

(IsHired ?d)

(decrease (number_of_openings_per_title ?t) 1)

(increase (Hired_count) 1)

(when (belongs-to-minority ?d) (increase (minority_applicants_hired) 1))

(when (belongs-to-majority ?d) (increase (majority_applicants_hired) 1)))

)

(:action **Mark-completion-of-hiring**

:parameters ( )

:precondition (and

(initialized)

(not(hiring_filled_all_titles))

)

:effect (and

(when (forall (?t - title) (and (=(number_of_openings_per_title ?t) 0))) (hiring_filled_all_titles)))

)

(:action **Initialize-after-hiring**

:parameters ( )

:precondition (and

(hiring_filled_all_titles)

(not(employee_project_counts_initialized))

)

:effect (and

(forall (?d - applicant) (when(isHired ?d) (assign (number_of_projects_assigned_to_employee ?d) 0)))

(employee_project_counts_initialized))

)

(:action **Assign-projects-round-1-diversity**

:parameters (?d - applicant ?t - title ?p - project)

:precondition (and

(employee_project_counts_initialized)

(>(titles_per_project ?t ?p) 0)

(isHired ?d)

(applicant_for_title ?d ?t)

(not(assigned_employee_to_project ?d ?p))

(not(exists (?c - applicant)

(and

(assigned_employee_to_project ?c ?p)

(exists (?r - race)(and(applicant-race ?c ?r)(applicant-race ?d ?r)))

(exists (?k - color)(and(applicant-color ?c ?k)(applicant-color ?d ?k)))

)))

)

:effect (and

(assigned_employee_to_project ?d ?p)

(increase (projects_diversity_count) 1)

(increase(number_of_projects_assigned_to_employee ?d) 1)

(decrease(titles_per_project ?t ?p) 1)

)

)

(:action **Preparing_for_Assigning_Round_2**

:parameters (?p - project)

```
:precondition (and  (employee_project_counts_initialized)

)

:effect (and

(forall (?d - applicant)

(when (and

(isHired ?d)

(not(qualified_for_round_2_assigning ?d ?p))

(not (assigned_employee_to_project ?d ?p))

(exists (?c - applicant)

(and

(IsHired ?c)

(assigned_employee_to_project ?c ?p)

(exists (?r - race)(and(applicant-race ?c ?r)(applicant-race ?d ?r)))

(exists (?k - color)(and(applicant-color ?c ?k)(applicant-color ?d ?k)))

))

)

(and
```

(qualified_for_round_2_assigning ?d ?p)

))

)

(when (and

(not (exists (?d - applicant)

(and

(IsHired ?d)

(not(assigned_employee_to_project ?d ?p))

(not(qualified_for_round_2_assigning ?d ?p))

)))

)

(and

(project_for_round_2 ?p)

))

))

(:action **Assign-projects-round-2**

:parameters (?d - applicant ?t - title ?p - project)

:precondition (and

(employee_project_counts_initialized)

(isHired ?d)

(not(assigned_employee_to_project ?d ?p))

(applicant_for_title ?d ?t)

(>(titles_per_project ?t ?p) 0)

(qualified_for_round_2_assigning ?d ?p)

(project_for_round_2 ?p)

)

:effect (and

(assigned_employee_to_project ?d ?p)

(increase(number_of_projects_assigned_to_employee ?d) 1)

(decrease(titles_per_project ?t ?p) 1)

))

(:action **Mark-completion-of-assigning**

:parameters ()

:precondition (and

(employee_project_counts_initialized)

(not(all-assigned))

)

:effect (and

;when every project is assigned with required count of titles

(when (forall(?t - title ?p - project)(and(=(titles_per_project ?t ?p)0))) (all-assigned))

))

)

Sample problem instance:

(define (problem P5) (:domain V2_software)

(:objects

a1 a2 a3 a4 a5 a6 a7 a8 a9 a10 - applicant

SE Manager - title

p1 p2 - project

Asian European-American-Non-Hispanic African-American American-Indian - race

white black brown - color

male female transgender - gender

straight gay - sexual-orientation

disabled none - disabilities

)

(:init

(applicant-race a1 Asian)(applicant-so a1 straight)

(applicant-gender a1 female)(applicant-disabilities a1 none)

(applicant-color a1 white)(applicant-race a2 Asian)

(applicant-so a2 straight)(applicant-gender a2 female)

(applicant-disabilities a2 none)(applicant-color a2 white)

(applicant-race a3 Asian)(applicant-so a3 straight)

(applicant-gender a3 male)(applicant-disabilities a3 disabled)

(applicant-color a3 white)(applicant-race a4 African-American)

(applicant-so a4 straight)(applicant-gender a4 male)

(applicant-disabilities a4 none)(applicant-color a4 black)

(applicant-race a5 American-Indian)(applicant-so a5 straight)

(applicant-gender a5 male)(applicant-disabilities a5 none)

(applicant-color a5 brown)(applicant-race a6 African-American)

(applicant-so a6 straight)(applicant-gender a6 female)

(applicant-disabilities a6 none)(applicant-color a6 black)

(applicant-race a7 European-American-Non-Hispanic)(applicant-so a7 straight)

(applicant-gender a7 female)(applicant-disabilities a7 none)

(applicant-color a7 white)(applicant-race a8 European-American-Non-Hispanic)

(applicant-so a8 straight)(applicant-gender a8 male)

(applicant-disabilities a8 disabled)(applicant-color a8 white)

(applicant-race a9 Asian)(applicant-so a9 straight)

(applicant-gender a9 male)(applicant-disabilities a9 none)

(applicant-color a9 white)(applicant-race a10 European-American-Non-Hispanic)

(applicant-so a10 straight)(applicant-gender a10 male)

(applicant-disabilities a10 none)(applicant-color a10 white)

(=(applicants_count) 0)(=(projects_count) 0)

(=(minority_applicant_count) 0)(=(majority_applicant_count)0)

(=(minority_applicants_hired)0)(=(majority_applicants_hired)0)

(=(Hired_count)0)(=(hired_for_diversity_count) 0)

(=(projects_diversity_count) 0)

(=(titles_per_project SE p1) 3)(=(titles_per_project SE p2) 3)

(=(titles_per_project Manager p1) 1)(=(titles_per_project Manager p2) 1)

(=(number_of_openings_per_title SE) 5)(=(number_of_openings_per_title Manager) 2)

(=(minimum_experience_required_for_title SE) 3)

(=(minimum_experience_required_for_title Manager) 10)

(=(applicant_experience a1) 2)(=(applicant_experience a2) 4)

(=(applicant_experience a3) 3)(=(applicant_experience a4) 3)

(=(applicant_experience a5) 5)(=(applicant_experience a6) 5)

(=(applicant_experience a7) 3)(=(applicant_experience a8) 6)

(=(applicant_experience a9) 6)(=(applicant_experience a10) 6)

(applicant_for_title a1 SE)(hasMasters a1)

(applicant_for_title a2 SE)(hasBachelors a2)

(applicant_for_title a3 SE)(hasMasters a3)

(applicant_for_title a4 SE)(hasPhD a4)

(applicant_for_title a5 Manager)(hasPhD a5)

(applicant_for_title a6 SE)(hasMasters a6)

(applicant_for_title a7 SE)(hasBachelors a7)

(applicant_for_title a8 Manager)(hasPhD a8)

(applicant_for_title a9 Manager)(hasPhD a9)

(applicant_for_title a10 Manager)(hasPhD a10)

)

(:goal(and (all-assigned)))

)

Version 3:

Domain definition:

(define (domain V3_Software)

(:requirements :adl :typing :fluents :negative-preconditions :strips)

(:types

applicant title project

)

(:predicates

(IsHired ?d - applicant)

(applicant_for_title ?d - applicant ?t - title)

(hasBachelors ?d - applicant)

(hasMasters ?d - applicant)

(hasPhD ?d - applicant)

(initialized)

(assigned_employee_to_project ?d - applicant ?p - project)

(hiring_filled_all_titles)

(employee_project_counts_initialized)

all-assigned)

)

(:functions

(applicants_count)

(applicant_experience ?d - applicant)

(minimum_experience_required_for_title ?t - title)

(number_of_openings_per_title ?t - title)

(total_per_title_from_each_project ?t - title) ;total_per_title_from_each project

(projects_count)

(titles_per_project ?t - title ?p - project) ; how many SE, Managers per project

(number_of_projects_assigned_to_employee ?d - applicant)

(Hired_count)

(number_of_titles_for_hiring ?t - title)

)

;define actions here

(:action **Initialize**

:parameters ()

:precondition (and

(not(initialized))

)

:effect (and

(forall (?c - applicant) (and(increase (applicants_count) 1)))

(forall (?c - applicant) (when (hasMasters ?c) (increase (applicant_experience ?c) 2)))

forall (?c - applicant) (when (hasPhD ?c) (increase (applicant_experience ?c) 5)))

(forall (?p - project) (and (increase (projects_count) 1)))

(forall (?t - title) (and(assign (total_per_title_from_each_project ?t) 0)))

(initialized)

))

(:action **Hiring**

:parameters (?d - applicant ?t - title)

:precondition (and

(initialized)

(applicant_for_title ?d ?t)

(>(number_of_openings_per_title ?t) 0)

(not(IsHired ?d))

(>=(applicant_experience ?d)(minimum_experience_required_for_title ?t))

)

:effect (and

(IsHired ?d)

(decrease (number_of_openings_per_title ?t) 1)

(increase (Hired_count) 1)

)

)

(:action **Mark-completion-of-hiring**

:parameters ( )

:precondition (and

(initialized)

(not(hiring_filled_all_titles))

)

:effect (and

(when (forall (?t - title) (and (=(number_of_openings_per_title ?t) 0))) (hiring_filled_all_titles))

)

)

(:action **Initialize-after-hiring**

:parameters ( )

:precondition (and

(hiring_filled_all_titles)

(not(employee_project_counts_initialized))

)

:effect (and

(forall (?d - applicant) (when(isHired ?d) (assign (number_of_projects_assigned_to_employee ?d) 0))

)

(forall (?p - project ?t - title) (and(increase (total_per_title_from_each_project ?t)

(titles_per_project ?t ?p))))

(employee_project_counts_initialized)

))

(:action **Assign-projects**

parameters (?d - applicant ?t - title ?p - project)

:precondition (and

(employee_project_counts_initialized)

(isHired ?d)

(not(assigned_employee_to_project ?d ?p))

(applicant_for_title ?d ?t)

(>(titles_per_project ?t ?p) 0)

(<(number_of_projects_assigned_to_employee ?d) (/(total_per_title_from_each_project ?t

)(number_of_titles_for_hiring ?t)))

)

:effect (and

(assigned_employee_to_project ?d ?p)

(increase(number_of_projects_assigned_to_employee ?d) 1)

(decrease(titles_per_project ?t ?p) 1)

)

)

(:action **Mark-completion-of-assigning**

:parameters ()

:precondition (and

(not(all-assigned))

(forall (?t - title) (not (exists (?d -applicant)(and(applicant_for_title ?d ?t)(isHired ?d)

(<(number_of_projects_assigned_to_employee ?d) (-(/(total_per_title_from_each_project ?t

)(number_of_titles_for_hiring ?t))1))))))

)

:effect (and

;when every project is assigned with required count of titles

when (forall(?t - title ?p - project)(and(=(titles_per_project ?t ?p)0))) (all-assigned))

)))

Sample problem instance:

(define (problem P5) (:domain V3_software)

(:objects

a1 a2 a3 a4 a5 a6 a7 a8 a9 a10 - applicant

SE Manager - title

p1 p2 - project

)

(:init

(=(Hired_count)0)(=(applicants_count) 0)

(=(projects_count) 0)(=(titles_per_project SE p1) 3)

(=(titles_per_project SE p2) 3)(=(titles_per_project Manager p1) 1)

(=(titles_per_project Manager p2) 1)(=(number_of_openings_per_title SE) 5)

(=(number_of_openings_per_title Manager) 2)(=(minimum_experience_required_for_title SE) 3)

(=(minimum_experience_required_for_title Manager) 10)

(=(number_of_titles_for_hiring SE) 5)(=(number_of_titles_for_hiring Manager) 2)

(=(applicant_experience a1) 2)(=(applicant_experience a2) 4)

(=(applicant_experience a3) 3)(=(applicant_experience a4) 3)

(=(applicant_experience a5) 5)(=(applicant_experience a6) 5)

(=(applicant_experience a7) 3)(=(applicant_experience a8) 6)

(=(applicant_experience a9) 6)(=(applicant_experience a10) 6)

(applicant_for_title a1 SE)(hasMasters a1)

(applicant_for_title a2 SE)(hasBachelors a2)

(applicant_for_title a3 SE)(hasMasters a3)

(applicant_for_title a4 SE)(hasPhD a4)

(applicant_for_title a5 Manager)(hasPhD a5)

(applicant_for_title a6 SE)(hasMasters a6)

(applicant_for_title a7 SE)(hasBachelors a7)

(applicant_for_title a8 Manager)(hasPhD a8)

(applicant_for_title a9 Manager)(hasPhD a9)

(applicant_for_title a10 Manager)(hasPhD a10)

)

(:goal (and(all-assigned))

))

Version 4:

Domain definition:

(define (domain V4_Software)

(:types

applicant title project

gender race sexual-orientation

disabilities color

)

(:predicates

(IsHired ?d - applicant)

(applicant_for_title ?d - applicant ?t - title)

(hasBachelors ?d - applicant)

(hasMasters ?d - applicant)

(hasPhD ?d - applicant)

(hiring_filled_all_titles)

(initialized)

(isWhite ?d - applicant)

(isTransgender ?d - applicant)

(isEuropean-American-Non-Hispanic ?d - applicant)

(isNotStraight ?d - applicant)

(isDisabled ?d - applicant)

(belongs-to-majority ?d - applicant)

(belongs-to-minority ?d - applicant)

(applicant-race ?d - applicant ?r - race)

(applicant-gender ?d - applicant ?g - gender)

(applicant-color ?d - applicant ?c - color)

(applicant-so ?d - applicant ?so - sexual-orientation)

(applicant-disabilities ?d - applicant ?ds - disabilities)

(qualified_for_hiring_round_2 ?d - applicant)

(can_hire_in_round_2 ?t - title)

(employee_project_counts_initialized)

(assigned_employee_to_project ?d - applicant ?p - project)

(all-assigned)

(qualified_for_round_2_assigning ?d - applicant ?p - project)

(project_for_round_2 ?p - project)

)

(:functions

(projects_diversity_count)

(number_of_titles_for_hiring ?t - title)

(titles_per_project ?t - title ?p - project)

(number_of_projects_assigned_to_employee ?d - applicant)

(hired_for_diversity_count)

(applicants_count)

(applicant_experience ?d - applicant)

(minimum_experience_required_for_title ?t - title)

(number_of_openings_per_title ?t - title)

(projects_count)

(Hired_count)

(total_per_title_from_each_project ?t - title)

(minority_applicant_count)

(majority_applicant_count)

(minority_applicants_hired)

(majority_applicants_hired)

)

(:action **Initialize**

:parameters ()

```
:precondition (and
(not(initialized))
)
:effect (and
(forall (?c - applicant) (and(increase (applicants_count) 1)))
(forall (?c - applicant) (when (hasMasters ?c) (increase (applicant_experience ?c) 2) ))
(forall (?c - applicant) (when (hasPhD ?c) (increase (applicant_experience ?c) 5) ))
(forall (?p - project) (and (increase (projects_count) 1)))
(forall (?t - title) (and(assign (total_per_title_from_each_project ?t) 0)))
(forall (?d - applicant) (when (and (isWhite ?d)(not(isNotStraight ?d))(not(isTransgender
?d))(isEuropean-American-Non-Hispanic ?d)(not (isDisabled ?d)))
(and (belongs-to-majority ?d)(increase (majority_applicant_count) 1))
)
)
(forall (?d - applicant) (when (or(isNotStraight ?d)(isDisabled ?d)(isTransgender
?d)(not(and(isWhite ?d)(isEuropean-American-Non-Hispanic ?d)))) ; Asians can be white
(and (belongs-to-minority ?d)(increase (minority_applicant_count) 1))
) )
```

(initialized)

))

(:action **Hiring_Round_1_Diversity**

:parameters (?d - applicant ?t - title)

:precondition (and

(initialized)

(>(number_of_openings_per_title ?t) 0)

(applicant_for_title ?d ?t)

(>=(applicant_experience ?d)(minimum_experience_required_for_title ?t))

(not(IsHired ?d))

(not(exists (?c - applicant)

(and

(IsHired ?c)

(applicant_for_title ?c ?t)

(exists (?r - race)(and(applicant-race ?c ?r)(applicant-race ?d ?r)))

(exists (?k - color)(and(applicant-color ?c ?k)(applicant-color ?d ?k)))

)))

)

:effect (and

(IsHired ?d)

(decrease (number_of_openings_per_title ?t) 1)

(increase (Hired_count) 1)

(increase (hired_for_diversity_count) 1)

(when (belongs-to-minority ?d) (increase (minority_applicants_hired) 1))

(when (belongs-to-majority ?d) (increase (majority_applicants_hired) 1))

))

(:action **Preparing_for_Hiring_Round_2**

:parameters ()

:precondition (and

(initialized)

)

:effect (and

(forall (?d - applicant)

(when (and

```
(not (IsHired ?d))

(not(qualified_for_hiring_round_2 ?d))

(exists (?c - applicant)

(and

(IsHired ?c)

(not(exists (?r - race)(and(applicant-race ?c ?r)(applicant-race ?d ?r))))

(not(exists (?k - color)(and(applicant-color ?c ?k)(applicant-color ?d ?k))))

) ))

(and

(qualified_for_hiring_round_2 ?d)

) ) )

(forall (?t - title)

(when ( and

(not (exists (?d - applicant)

(and

(applicant_for_title ?d ?t)

(not(IsHired ?d))
```

(not(qualified_for_hiring_round_2 ?d))

)))

)

(and

(can_hire_in_round_2 ?t)

))))

)

(:action **Hiring_Round_2**

:parameters (?d - applicant ?t - title)

:precondition (and

(qualified_for_hiring_round_2 ?d)

(>(number_of_openings_per_title ?t) 0)

(applicant_for_title ?d ?t)

(can_hire_in_round_2 ?t)

(>=(applicant_experience ?d)(minimum_experience_required_for_title ?t))

(not(IsHired ?d))

)

:effect (and

(IsHired ?d)

(decrease (number_of_openings_per_title ?t) 1)

(increase (Hired_count) 1)

(when (belongs-to-minority ?d) (increase (minority_applicants_hired) 1))

(when (belongs-to-majority ?d) (increase (majority_applicants_hired) 1))

)

)

(:action **Mark-completion-of-hiring**

:parameters ( )

:precondition (and

(initialized)

(not(hiring_filled_all_titles))

)

:effect (and

(when (forall (?t - title) (and (=(number_of_openings_per_title ?t) 0))) (hiring_filled_all_titles))

(forall (?p - project ?t - title) (and(increase (total_per_title_from_each_project ?t)

(titles_per_project ?t ?p))))

)

)

(:action **Initialize-after-hiring**

:parameters ( )

:precondition (and

(hiring_filled_all_titles)

(not(employee_project_counts_initialized))

)

:effect (and

(forall (?d - applicant) (when(isHired ?d) (assign (number_of_projects_assigned_to_employee ?d) 0))

)(employee_project_counts_initialized))

)

(:action **Assign-projects-round-1-diversity**

:parameters (?d - applicant ?t - title ?p - project)

:precondition (and

(employee_project_counts_initialized)

(>(titles_per_project ?t ?p) 0)

```
(isHired ?d)

(applicant_for_title ?d ?t)

(not(assigned_employee_to_project ?d ?p))

(<(number_of_projects_assigned_to_employee ?d) (/(total_per_title_from_each_project ?t

)(number_of_titles_for_hiring ?t) ))

(not(exists (?c - applicant)

(and

(assigned_employee_to_project ?c ?p)

(exists (?r - race)(and(applicant-race ?c ?r)(applicant-race ?d ?r)))

(exists (?k - color)(and(applicant-color ?c ?k)(applicant-color ?d ?k)))

) ) )

)

:effect (and

(assigned_employee_to_project ?d ?p)

(increase (projects_diversity_count) 1)

(increase(number_of_projects_assigned_to_employee ?d) 1)

(decrease(titles_per_project ?t ?p) 1)

)
```

)

(:action **Preparing_for_Assigning_Round_2**

:parameters (?p - project)

:precondition (and (employee_project_counts_initialized)

)

:effect (and

(forall (?d - applicant)

(when (and

(isHired ?d)

(not(qualified_for_round_2_assigning ?d ?p))

(not (assigned_employee_to_project ?d ?p))

(exists (?c - applicant)

(and

(IsHired ?c)

(assigned_employee_to_project ?c ?p)

(exists (?r - race)(and(applicant-race ?c ?r)(applicant-race ?d ?r)))

(exists (?k - color)(and(applicant-color ?c ?k)(applicant-color ?d ?k)))

```
)

))

(and

(qualified_for_round_2_assigning ?d ?p)

) )

)

(when ( and

(not (exists (?d - applicant)

(and

(IsHired ?d)

(not(assigned_employee_to_project ?d ?p))

(not(qualified_for_round_2_assigning ?d ?p))

) )

) )

(and

(project_for_round_2 ?p)

)
```

)))

(:action **Assign-projects-round-2**

:parameters (?d - applicant ?t - title ?p - project)

:precondition (and

(employee_project_counts_initialized)

(isHired ?d)

(not(assigned_employee_to_project ?d ?p))

(applicant_for_title ?d ?t)

(>(titles_per_project ?t ?p) 0)

(<(number_of_projects_assigned_to_employee ?d) (/(total_per_title_from_each_project ?t)(number_of_titles_for_hiring ?t)))

(qualified_for_round_2_assigning ?d ?p)

(project_for_round_2 ?p)

)

:effect (and

(assigned_employee_to_project ?d ?p)

(increase(number_of_projects_assigned_to_employee ?d) 1)

(decrease(titles_per_project ?t ?p) 1)

)

)

(:action **Mark-completion-of-assigning**

:parameters ()

:precondition (and

(employee_project_counts_initialized)

(not(all-assigned))

(forall (?t - title) (not (exists (?d -applicant)(and(applicant_for_title ?d ?t)(isHired ?d)

(<(number_of_projects_assigned_to_employee ?d) (-(/(total_per_title_from_each_project ?t

)(number_of_titles_for_hiring ?t))1))))))

)

:effect (and

(when (forall(?t - title ?p - project)(and(=(titles_per_project ?t ?p)0))) (all-assigned))

)))

Sample problem instance:

(define (problem P4) (:domain V4_software)

(:objects

a1 a2 a3 a4 a5 a6 a7 a8 a9 - applicant

SE Manager - title

p1 p2 - project

Asian European-American-Non-Hispanic African-American American-Indian - race

white black brown - color

male female transgender - gender

straight gay - sexual-orientation

disabled none - disabilities

)

(:init

(applicant-race a1 Asian)(applicant-so a1 straight)

(applicant-gender a1 female)(applicant-disabilities a1 none)

(applicant-color a1 brown)(applicant-race a2 Asian)

(applicant-so a2 straight)(applicant-gender a2 female)

(applicant-disabilities a2 none)(applicant-color a2 white)

(applicant-race a3 Asian)(applicant-so a3 straight)

(applicant-gender a3 female)(applicant-disabilities a3 none)

(applicant-color a3 white)(applicant-race a4 African-American)

(applicant-so a4 straight)(applicant-gender a4 male)

(applicant-disabilities a4 none)(applicant-color a4 black)

(applicant-race a5 European-American-Non-Hispanic)

(applicant-so a5 straight)(applicant-gender a5 male)

(applicant-disabilities a5 none)(applicant-color a5 white)

(applicant-race a6 European-American-Non-Hispanic)

(applicant-so a6 straight)(applicant-gender a6 male)

(applicant-disabilities a6 none)(applicant-color a6 white)

(applicant-race a7 European-American-Non-Hispanic)

(applicant-so a7 straight)(applicant-gender a7 female)

(applicant-disabilities a7 none)(applicant-color a7 white)

(applicant-race a8 European-American-Non-Hispanic)

(applicant-so a8 straight)(applicant-gender a8 male)

(applicant-disabilities a8 disabled)(applicant-color a8 white)

(applicant-race a9 European-American-Non-Hispanic)

(applicant-so a9 straight)(applicant-gender a9 male)

(applicant-disabilities a9 none)(applicant-color a9 white)

(isWhite a2)(isWhite a3)(isWhite a5)(isWhite a7)(isWhite a6)(isWhite a8)(isWhite a9)

(isEuropean-American-Non-Hispanic a2) (isEuropean-American-Non-Hispanic a3)(isEuropean-American-Non-Hispanic a5)(isEuropean-American-Non-Hispanic a6)

(isEuropean-American-Non-Hispanic a7)(isEuropean-American-Non-Hispanic a8)(isEuropean-American-Non-Hispanic a9)

(isDisabled a8)

(=(hired_for_diversity_count) 0)(=(projects_diversity_count) 0)

(=(applicants_count) 0)(=(projects_count) 0)

(=(minority_applicant_count) 0)(=(majority_applicant_count)0)

(=(minority_applicants_hired)0)(=(majority_applicants_hired)0)

(=(Hired_count)0)(=(titles_per_project SE p1) 2)

(=(titles_per_project SE p2) 3)(=(titles_per_project Manager p1) 1)

(=(titles_per_project Manager p2) 1)(=(number_of_openings_per_title SE) 3)

(=(number_of_openings_per_title Manager) 2)(=(minimum_experience_required_for_title SE) 3)

(=(minimum_experience_required_for_title Manager) 10)

(=(number_of_titles_for_hiring SE) 3)

(=(number_of_titles_for_hiring Manager) 2)

(=(applicant_experience a1) 2)(=(applicant_experience a2) 4)

(=(applicant_experience a3) 3)(=(applicant_experience a4) 3)

(=(applicant_experience a5) 5)(=(applicant_experience a6) 5)

(=(applicant_experience a7) 3)(=(applicant_experience a8) 6)

(=(applicant_experience a9) 6)

(applicant_for_title a1 SE)(hasMasters a1)

(applicant_for_title a2 SE)(hasBachelors a2)

(applicant_for_title a3 SE)(hasMasters a3)

(applicant_for_title a4 SE)(hasPhD a4)

(applicant_for_title a5 Manager)(hasPhD a5)

(applicant_for_title a6 SE)(hasMasters a6)

(applicant_for_title a7 SE)(hasBachelors a7)

(applicant_for_title a8 Manager)(hasPhD a8)

(applicant_for_title a9 Manager)(hasPhD a9)

)

(:goal (and(all-assigned)))

)

Version 5:

Domain definition:

(define (domain V5_Software)

(:types

applicant title project gender

race sexual-orientation disabilities color

)

(:predicates

(IsHired ?d - applicant) (applicant_for_title ?d - applicant ?t - title)

(hasBachelors ?d – applicant)(hasMasters ?d - applicant)

(hasPhD ?d – applicant) (hiring_filled_all_titles)

(initialized) (isWhite ?d - applicant) (isTransgender ?d - applicant)

(isEuropean-American-Non-Hispanic ?d - applicant)

(isNotStraight ?d - applicant) (isDisabled ?d - applicant)

(belongs-to-majority ?d - applicant) (belongs-to-minority ?d - applicant)

(applicant-race ?d - applicant ?r - race) (applicant-gender ?d - applicant ?g - gender)

(applicant-color ?d - applicant ?c - color) (applicant-so ?d - applicant ?so - sexual-orientation)

(applicant-disabilities ?d - applicant ?ds - disabilities) (qualified_for_hiring_round_2 ?d - applicant)(can_hire_in_round_2 ?t - title) (employee_project_counts_initialized)

(assigned_employee_to_project ?d - applicant ?p - project) (all-assigned)

(qualified_for_round_2_assigning ?d - applicant ?p - project) (project_for_round_2 ?p - project)

)

(:functions

(projects_diversity_count) (number_of_titles_for_hiring ?t - title)

(titles_per_project ?t - title ?p - project)

(number_of_projects_assigned_to_employee ?d - applicant)

(hired_for_diversity_count) (applicants_count)

(applicant_experience ?d - applicant)

(minimum_experience_required_for_title ?t - title)

(number_of_openings_per_title ?t - title)

(projects_count) (Hired_count)

(total_per_title_from_each_project ?t - title)

(minority_applicant_count) (majority_applicant_count)

(minority_applicants_hired) (majority_applicants_hired)

)

(:action **Initialize**

:parameters ()

:precondition (and

(not(initialized))

)

:effect (and

(forall (?c - applicant) (and(increase (applicants_count) 1)))

(forall (?c - applicant) (when (hasMasters ?c) (increase (applicant_experience ?c) 2)))

(forall (?c - applicant) (when (hasPhD ?c) (increase (applicant_experience ?c) 5)))

(forall (?p - project) (and (increase (projects_count) 1)))

(forall (?t - title) (and(assign (total_per_title_from_each_project ?t) 0)))

(forall (?d - applicant) (when (and (isWhite ?d)(not(isNotStraight ?d))(not(isTransgender ?d))(isEuropean-American-Non-Hispanic ?d)(not (isDisabled ?d)))

(and (belongs-to-majority ?d)(increase (majority_applicant_count) 1))

))

(forall (?d - applicant) (when (or(isNotStraight ?d)(isDisabled ?d)(isTransgender ?d)(not(and(isWhite ?d)(isEuropean-American-Non-Hispanic ?d)))) ; Asians can be white

(and (belongs-to-minority ?d)(increase (minority_applicant_count) 1))

))

(initialized)

))

(:action **Hiring_Round_1_Diversity**

:parameters (?d - applicant ?t - title)

:precondition (and

(initialized)

(>(number_of_openings_per_title ?t) 0)

(applicant_for_title ?d ?t)

(>=(applicant_experience ?d)(minimum_experience_required_for_title ?t))

(not(IsHired ?d))

(not(exists (?c - applicant)

(and

(IsHired ?c)

(applicant_for_title ?c ?t)

(exists (?r - race)(and(applicant-race ?c ?r)(applicant-race ?d ?r)))

(exists (?k - color)(and(applicant-color ?c ?k)(applicant-color ?d ?k)))

(exists (?g - gender)(and(applicant-gender ?c ?g)(applicant-gender ?d ?g)))

(exists (?x - sexual-orientation)(and(applicant-so ?c ?x)(applicant-so ?d ?x)))

(exists (?y - disabilities)(and(applicant-disabilities ?c ?y)(applicant-disabilities ?d ?y)))

))))

:effect (and

(IsHired ?d)

(decrease (number_of_openings_per_title ?t) 1)

(increase (Hired_count) 1)

(increase (hired_for_diversity_count) 1)

(when (belongs-to-minority ?d) (increase (minority_applicants_hired) 1))

(when (belongs-to-majority ?d) (increase (majority_applicants_hired) 1))

))

(:action **Preparing_for_Hiring_Round_2**

:parameters ()

:precondition (and

(initialized)

)

```
:effect (and

(forall (?d - applicant)

(when ( and

(not (IsHired ?d))

(not(qualified_for_hiring_round_2 ?d))

(exists (?c - applicant)

(and

(IsHired ?c)

(not(exists (?r - race)(and(applicant-race ?c ?r)(applicant-race ?d ?r))))

(not(exists (?k - color)(and(applicant-color ?c ?k)(applicant-color ?d ?k))))

(not(exists (?g - gender)(and(applicant-gender ?c ?g)(applicant-gender ?d ?g))))

(not(exists (?x - sexual-orientation)(and(applicant-so ?c ?x)(applicant-so ?d ?x))))

(not(exists (?y - disabilities)(and(applicant-disabilities ?c ?y)(applicant-disabilities ?d ?y))))

) ) )

(and

(qualified_for_hiring_round_2 ?d)

) ) )
```

(forall (?t - title)

(when (and

(not (exists (?d - applicant)

(and

(applicant_for_title ?d ?t)

(not(IsHired ?d))

(not(qualified_for_hiring_round_2 ?d))

)))

)

(and

(can_hire_in_round_2 ?t)

))))

)

(:action **Hiring_Round_2**

:parameters (?d - applicant ?t - title)

:precondition (and

(qualified_for_hiring_round_2 ?d)

(>(number_of_openings_per_title ?t) 0)

(applicant_for_title ?d ?t)

(can_hire_in_round_2 ?t)

(>=(applicant_experience ?d)(minimum_experience_required_for_title ?t))

(not(IsHired ?d))

;there does not exist an applicant who is qualifying,not hired, and has different race or color to that of ?D

)

:effect (and

(IsHired ?d)

(decrease (number_of_openings_per_title ?t) 1)

(increase (Hired_count) 1)

(when (belongs-to-minority ?d) (increase (minority_applicants_hired) 1))

(when (belongs-to-majority ?d) (increase (majority_applicants_hired) 1))

)

)

(:action **Mark-completion-of-hiring**

:parameters ( )

:precondition (and

(initialized)

(not(hiring_filled_all_titles))

)

:effect (and

(when (forall (?t - title) (and (=(number_of_openings_per_title ?t) 0))) (hiring_filled_all_titles))

(forall (?p - project ?t - title) (and(increase (total_per_title_from_each_project ?t)

(titles_per_project ?t ?p))))

)

)

(:action **Initialize-after-hiring**

:parameters ( )

:precondition (and

(hiring_filled_all_titles)

(not(employee_project_counts_initialized))

)

:effect (and

(forall (?d - applicant) (when(isHired ?d) (assign (number_of_projects_assigned_to_employee ?d) 0))

)

(employee_project_counts_initialized))

)

(:action **Assign-projects-round-1-diversity**

:parameters (?d - applicant ?t - title ?p - project)

:precondition (and

(employee_project_counts_initialized)

(>(titles_per_project ?t ?p) 0)

(isHired ?d)

(applicant_for_title ?d ?t)

(not(assigned_employee_to_project ?d ?p))

(<(number_of_projects_assigned_to_employee ?d) (/(total_per_title_from_each_project ?t)(number_of_titles_for_hiring ?t)))

(not(exists (?c - applicant)

(and

(assigned_employee_to_project ?c ?p)

(exists (?r - race)(and(applicant-race ?c ?r)(applicant-race ?d ?r)))

(exists (?k - color)(and(applicant-color ?c ?k)(applicant-color ?d ?k)))

(exists (?g - gender)(and(applicant-gender ?c ?g)(applicant-gender ?d ?g)))

(exists (?x - sexual-orientation)(and(applicant-so ?c ?x)(applicant-so ?d ?x)))

(exists (?y - disabilities)(and(applicant-disabilities ?c ?y)(applicant-disabilities ?d ?y)))

)))

)

:effect (and

(assigned_employee_to_project ?d ?p)

(increase (projects_diversity_count) 1)

(increase(number_of_projects_assigned_to_employee ?d) 1)

(decrease(titles_per_project ?t ?p) 1)

)

)

(:action **Preparing_for_Assigning_Round_2**

:parameters (?p - project)

:precondition (and (employee_project_counts_initialized)

)

:effect (and

(forall (?d - applicant)

(when (and

(isHired ?d)

(not(qualified_for_round_2_assigning ?d ?p))

(not (assigned_employee_to_project ?d ?p))

(exists (?c - applicant)

(and

(IsHired ?c)

(assigned_employee_to_project ?c ?p)

(exists (?r - race)(and(applicant-race ?c ?r)(applicant-race ?d ?r)))

(exists (?k - color)(and(applicant-color ?c ?k)(applicant-color ?d ?k)))

(exists (?g - gender)(and(applicant-gender ?c ?g)(applicant-gender ?d ?g)))

(exists (?x - sexual-orientation)(and(applicant-so ?c ?x)(applicant-so ?d ?x)))

(exists (?y - disabilities)(and(applicant-disabilities ?c ?y)(applicant-disabilities ?d ?y)))

))

)

(and

(qualified_for_round_2_assigning ?d ?p)

)))

(when (and

(not (exists (?d - applicant)

(and

(IsHired ?d)

(not(assigned_employee_to_project ?d ?p))

(not(qualified_for_round_2_assigning ?d ?p))

)))

)

(and

(project_for_round_2 ?p))

))

)

(:action **Assign-projects-round-2**

```
:parameters ( ?d - applicant ?t - title ?p - project)

:precondition (and

(employee_project_counts_initialized)

(isHired ?d)

(not(assigned_employee_to_project ?d ?p))

(applicant_for_title ?d ?t)

(>(titles_per_project ?t ?p ) 0)

(<(number_of_projects_assigned_to_employee ?d) (/(total_per_title_from_each_project ?t

)(number_of_titles_for_hiring ?t) ))

(qualified_for_round_2_assigning ?d ?p)

(project_for_round_2 ?p )

)

:effect (and

(assigned_employee_to_project ?d ?p)

(increase(number_of_projects_assigned_to_employee ?d) 1)

(decrease(titles_per_project ?t ?p) 1)

)

)
```

(:action **Mark-completion-of-assigning**

:parameters ()

:precondition (and

(employee_project_counts_initialized)

(not(all-assigned))

(forall (?t - title) (not (exists (?d -applicant)(and(applicant_for_title ?d ?t)(isHired ?d)

(<(number_of_projects_assigned_to_employee ?d) (-(/(total_per_title_from_each_project ?t

)(number_of_titles_for_hiring ?t))1))))))

)

:effect (and

;when every project is assigned with required count of titles

(when (forall(?t - title ?p - project)(and(=(titles_per_project ?t ?p)0))) (all-assigned))

)))

Sample problem instance:

(define (problem P4) (:domain V5_software)

(:objects

a1 a2 a3 a4 a5 a6 a7 a8 a9 - applicant

SE Manager - title

p1 p2 - project

Asian European-American-Non-Hispanic African-American American-Indian - race

white black brown - color

male female transgender - gender

straight gay - sexual-orientation

disabled none - disabilities

)

(:init

(applicant-race a1 Asian)(applicant-so a1 straight)

(applicant-gender a1 female)(applicant-disabilities a1 none)

(applicant-color a1 brown)(applicant-race a2 Asian)

(applicant-so a2 straight)(applicant-gender a2 female)

(applicant-disabilities a2 none)(applicant-color a2 white)

(applicant-race a3 Asian)(applicant-so a3 straight)

(applicant-gender a3 female)(applicant-disabilities a3 none)

(applicant-color a3 white)(applicant-race a4 African-American)

(applicant-so a4 straight)(applicant-gender a4 male)

(applicant-disabilities a4 none)(applicant-color a4 black)

(applicant-race a5 European-American-Non-Hispanic)

(applicant-so a5 straight)(applicant-gender a5 male)

(applicant-disabilities a5 none)(applicant-color a5 white)

(applicant-race a6 European-American-Non-Hispanic)

(applicant-so a6 straight)(applicant-gender a6 male)

(applicant-disabilities a6 none)(applicant-color a6 white)

(applicant-race a7 European-American-Non-Hispanic)

(applicant-so a7 straight)(applicant-gender a7 female)

(applicant-disabilities a7 none)(applicant-color a7 white)

(applicant-race a8 European-American-Non-Hispanic)

(applicant-so a8 straight)(applicant-gender a8 male)

(applicant-disabilities a8 disabled)(applicant-color a8 white)

(applicant-race a9 European-American-Non-Hispanic)

(applicant-so a9 straight)(applicant-gender a9 male)

(applicant-disabilities a9 none)(applicant-color a9 white)

(isWhite a2)(isWhite a3)(isWhite a5)(isWhite a7)(isWhite a6)(isWhite a8)(isWhite a9)

(isEuropean-American-Non-Hispanic a2) (isEuropean-American-Non-Hispanic a3)(isEuropean-American-Non-Hispanic a5)(isEuropean-American-Non-Hispanic a6)

(isEuropean-American-Non-Hispanic a7)(isEuropean-American-Non-Hispanic a8)(isEuropean-American-Non-Hispanic a9)

(isDisabled a8)

(=(hired_for_diversity_count) 0)(=(projects_diversity_count) 0)

(=(applicants_count) 0)(=(projects_count) 0)

(=(minority_applicant_count) 0)(=(majority_applicant_count)0)

(=(minority_applicants_hired)0)(=(majority_applicants_hired)0)

(=(Hired_count)0)(=(titles_per_project SE p1) 2)

(=(titles_per_project SE p2) 3)(=(titles_per_project Manager p1) 1)

(=(titles_per_project Manager p2) 1)(=(number_of_openings_per_title SE) 3)

(=(number_of_openings_per_title Manager) 2)(=(minimum_experience_required_for_title SE) 3)

(=(minimum_experience_required_for_title Manager) 10)

(=(number_of_titles_for_hiring SE) 3)

(=(number_of_titles_for_hiring Manager) 2)

(=(applicant_experience a1) 2)(=(applicant_experience a2) 4)

(=(applicant_experience a3) 3)(=(applicant_experience a4) 3)

(=(applicant_experience a5) 5)(=(applicant_experience a6) 5)

(=(applicant_experience a7) 3)(=(applicant_experience a8) 6)

(=(applicant_experience a9) 6)

(applicant_for_title a1 SE)(hasMasters a1)

(applicant_for_title a2 SE)(hasBachelors a2)

(applicant_for_title a3 SE)(hasMasters a3)

(applicant_for_title a4 SE)(hasPhD a4)

(applicant_for_title a5 Manager)(hasPhD a5)

(applicant_for_title a6 SE)(hasMasters a6)

(applicant_for_title a7 SE)(hasBachelors a7)

(applicant_for_title a8 Manager)(hasPhD a8)

(applicant_for_title a9 Manager)(hasPhD a9)

)

(:goal (and(all-assigned)))

(:metric maximize (+(diversity_count)(inclusivity_count))))

Version 6:

Domain definition:

```
(define (domain V6_Software)
(:requirements :adl :typing :fluents :negative-preconditions :strips)
(:types
applicant title project gender
race sexual-orientation disabilities
color
)
(:predicates
(IsHired ?d - applicant) (applicant_for_title ?d - applicant ?t - title)
(hasBachelors ?d - applicant) (hasMasters ?d - applicant)
(hasPhD ?d - applicant) (hiring_filled_all_titles)
(initialized) (isWhite ?d - applicant)
(isTransgender ?d - applicant) (isEuropean-American-Non-Hispanic ?d - applicant)
(isNotStraight ?d - applicant) (isDisabled ?d - applicant)
(belongs-to-majority ?d - applicant) (belongs-to-minority ?d - applicant)
(applicant-race ?d - applicant ?r - race) (applicant-gender ?d - applicant ?g - gender)
(applicant-color ?d - applicant ?c - color) (applicant-so ?d - applicant ?so - sexual-orientation)
```

(applicant-disabilities ?d - applicant ?ds - disabilities)

(qualified_for_hiring_round_2 ?d - applicant) (can_hire_in_round_2 ?t - title)

(employee_project_counts_initialized)

(assigned_employee_to_project ?d - applicant ?p - project)

(all-assigned) (qualified_for_assigning_round_2 ?d - applicant ?p - project)

(project_for_round_2 ?p - project) (counted)

(accepted)

)

(:functions

(k1)

(k2)

(minority_applicants_for_title ?t - title)(majority_applicants_for_title ?t - title)

(hired_minority_applicants_for_title ?t - title)

(hired_majority_applicants_for_title ?t - title)(projects_diversity_count)

(number_of_titles_for_hiring ?t - title)(titles_per_project ?t - title ?p - project)

(number_of_projects_assigned_to_employee ?d - applicant)

(hired_for_diversity_count)(applicants_count)

(applicant_experience ?d - applicant)(minimum_experience_required_for_title ?t - title)

(number_of_openings_per_title ?t - title)(projects_count)(Hired_count)

(total_per_title_from_each_project ?t - title)

(minority_applicant_count)(majority_applicant_count)

(minority_applicants_hired)(majority_applicants_hired)

)

(:action **Initialize**

:parameters ()

:precondition (and

(not(initialized))

)

:effect (and

(forall (?c - applicant) (and(increase (applicants_count) 1)))

(forall (?c - applicant) (when (hasMasters ?c) (increase (applicant_experience ?c) 2)))

(forall (?c - applicant) (when (hasPhD ?c) (increase (applicant_experience ?c) 5)))

(forall (?p - project) (and (increase (projects_count) 1)))

(forall (?t - title) (and

(assign (total_per_title_from_each_project ?t) 0)

(assign (hired_majority_applicants_for_title ?t) 0)

(assign (hired_minority_applicants_for_title ?t) 0)

))

(forall (?d - applicant) (when (and (isWhite ?d)(isEuropean-American-Non-Hispanic ?d)(not(or (isTransgender ?d) (isDisabled ?d))))

(and (belongs-to-majority ?d)(increase (majority_applicant_count) 1))

)

)

(forall (?d - applicant) (when (or (isDisabled ?d)(isTransgender ?d)(not(and(isWhite ?d)(isEuropean-American-Non-Hispanic ?d)))) ; Asians can be white

(and (belongs-to-minority ?d)(increase (minority_applicant_count) 1))

))

(initialized)

))

(:action **count_minority_majority_applicants**

:parameters ()

:precondition (and

(initialized)

(not(counted))

)

:effect (and

(forall (?d - applicant ?t - title) (when (and (belongs-to-minority ?d)(applicant_for_title ?d ?t)(>=(applicant_experience ?d)(minimum_experience_required_for_title ?t)))

(and (increase (minority_applicants_for_title ?t) 1))

)

)

(forall (?d - applicant ?t - title) (when (and (belongs-to-majority ?d)(applicant_for_title ?d ?t)(>=(applicant_experience ?d)(minimum_experience_required_for_title ?t)))

(and (increase (majority_applicants_for_title ?t) 1))

))

(counted))

)

(:action **Hiring_Round_1_Diversity**

:parameters (?d - applicant ?t - title)

:precondition (and

(counted)

(>(number_of_openings_per_title ?t) 0)

(applicant_for_title ?d ?t)

(>=(applicant_experience ?d)(minimum_experience_required_for_title ?t))

(not(IsHired ?d))

(not(exists (?c - applicant)

(and

(IsHired ?c)

(applicant_for_title ?c ?t)

(exists (?r - race)(and(applicant-race ?c ?r)(applicant-race ?d ?r)))

(exists (?k - color)(and(applicant-color ?c ?k)(applicant-color ?d ?k)))

(exists (?g - gender)(and(applicant-gender ?c ?g)(applicant-gender ?d ?g)))

(exists (?x - sexual-orientation)(and(applicant-so ?c ?x)(applicant-so ?d ?x)))

(exists (?y - disabilities)(and(applicant-disabilities ?c ?y)(applicant-disabilities ?d ?y)))

))))

:effect (and

(IsHired ?d)

(decrease (number_of_openings_per_title ?t) 1)

(increase (Hired_count) 1)

(increase (hired_for_diversity_count) 1)

(when (belongs-to-minority ?d) (and (increase (hired_minority_applicants_for_title ?t) 1)(increase (minority_applicants_hired) 1)))

(when (belongs-to-majority ?d) (and (increase (hired_majority_applicants_for_title ?t) 1)(increase (majority_applicants_hired) 1)))

))

(:action **Preparing_for_Hiring_Round_2**

:parameters ()

:precondition (and

(initialized)

(counted)

)

:effect (and

(forall (?d - applicant)

(when (and

(not (IsHired ?d))

```
(not(qualified_for_hiring_round_2 ?d))
(exists (?c - applicant)
(and
(IsHired ?c)
(or
(not(exists (?r - race)(and(applicant-race ?c ?r)(applicant-race ?d ?r))))
(not(exists (?k - color)(and(applicant-color ?c ?k)(applicant-color ?d ?k))))
(not(exists (?g - gender)(and(applicant-gender ?c ?g)(applicant-gender ?d ?g))))
(not(exists (?x - sexual-orientation)(and(applicant-so ?c ?x)(applicant-so ?d ?x))))
(not(exists (?y - disabilities)(and(applicant-disabilities ?c ?y)(applicant-disabilities ?d ?y))))
) ) ) )
(and
(qualified_for_hiring_round_2 ?d)
) ) )
(forall (?t - title)
(when ( and
(not (exists (?d - applicant)
```

(and

(applicant_for_title ?d ?t)

(not(IsHired ?d))

(not(qualified_for_hiring_round_2 ?d))

))))

(and

(can_hire_in_round_2 ?t)

))))

)

(:action **Hiring_Round_2**

:parameters (?d - applicant ?t - title)

:precondition (and

(qualified_for_hiring_round_2 ?d)

(>(number_of_openings_per_title ?t) 0)

(applicant_for_title ?d ?t)

(can_hire_in_round_2 ?t)

(>=(applicant_experience ?d)(minimum_experience_required_for_title ?t))

(not(IsHired ?d))

;there does not exist an applicant who is qualifying,not hired, and has different race or color to that of ?D

)

:effect (and

(IsHired ?d)

(decrease (number_of_openings_per_title ?t) 1)

(increase (Hired_count) 1)

(when (belongs-to-minority ?d) (and (increase (hired_minority_applicants_for_title ?t) 1)(increase (minority_applicants_hired) 1)))

(when (belongs-to-majority ?d) (and (increase (hired_majority_applicants_for_title ?t) 1)(increase (majority_applicants_hired) 1)))

)

)

(:action **Mark-completion-of-hiring**

:parameters ( )

:precondition (and

(initialized)

(not(hiring_filled_all_titles))

)

:effect (and

(when (forall (?t - title) (and (=(number_of_openings_per_title ?t) 0))) (hiring_filled_all_titles))

(forall (?p - project ?t - title) (and(increase (total_per_title_from_each_project ?t)

(titles_per_project ?t ?p))))

)

)

(:action **Mark-completion-of-Checking-Acceptance**

:parameters ( )

:precondition (and

(not(accepted))

(hiring_filled_all_titles)

(not (exists (?t - title) (and

(<= (+ (*(k1)(minority_applicants_for_title ?t)) (*(k2)(majority_applicants_for_title

?t)))(number_of_titles_for_hiring ?t))

(or

(>(- (*(k1)(minority_applicants_for_title ?t)) 1)(hired_minority_applicants_for_title ?t))

(>(- (*(k2)(majority_applicants_for_title ?t)) 1)(hired_majority_applicants_for_title ?t))

))))

)

:effect (and

(accepted)

))

(:action **Initialize-after-hiring**

:parameters ( )

:precondition (and

(accepted)

(not(employee_project_counts_initialized))

)

:effect (and

(forall (?d - applicant) (when(isHired ?d) (assign (number_of_projects_assigned_to_employee ?d) 0))

) (employee_project_counts_initialized)

))

(:action **Assign-projects-round-1-diversity**

:parameters (?d - applicant ?t - title ?p - project)

:precondition (and

(employee_project_counts_initialized)

(>(titles_per_project ?t ?p) 0)

(isHired ?d)

(applicant_for_title ?d ?t)

(not(assigned_employee_to_project ?d ?p))

(<(number_of_projects_assigned_to_employee ?d) (/(total_per_title_from_each_project ?t)(number_of_titles_for_hiring ?t)))

(not(exists (?c - applicant)

(and

(assigned_employee_to_project ?c ?p)

(exists (?r - race)(and(applicant-race ?c ?r)(applicant-race ?d ?r)))

(exists (?k - color)(and(applicant-color ?c ?k)(applicant-color ?d ?k)))

(exists (?g - gender)(and(applicant-gender ?c ?g)(applicant-gender ?d ?g)))

(exists (?x - sexual-orientation)(and(applicant-so ?c ?x)(applicant-so ?d ?x)))

(exists (?y - disabilities)(and(applicant-disabilities ?c ?y)(applicant-disabilities ?d ?y)))

))))

:effect (and

(assigned_employee_to_project ?d ?p)

(increase (projects_diversity_count) 1)

(increase(number_of_projects_assigned_to_employee ?d) 1)

(decrease(titles_per_project ?t ?p) 1)

))

(:action **Preparing_for_Assigning_Round_2**

:parameters (?p - project)

:precondition (and (employee_project_counts_initialized)

)

:effect (and

(forall (?d - applicant)

(when (and

(isHired ?d)

(not(qualified_for_assigning_round_2 ?d ?p))

(not (assigned_employee_to_project ?d ?p))

(exists (?c - applicant)

```
(and
(IsHired ?c)
(assigned_employee_to_project ?c ?p)
(exists (?r - race)(and(applicant-race ?c ?r)(applicant-race ?d ?r)))
(exists (?k - color)(and(applicant-color ?c ?k)(applicant-color ?d ?k)))
(exists (?g - gender)(and(applicant-gender ?c ?g)(applicant-gender ?d ?g)))
(exists (?x - sexual-orientation)(and(applicant-so ?c ?x)(applicant-so ?d ?x)))
(exists (?y - disabilities)(and(applicant-disabilities ?c ?y)(applicant-disabilities ?d ?y)))
)))
(and
(qualified_for_assigning_round_2 ?d ?p) ) ) )
(when ( and
(not (exists (?d - applicant)
(and
(IsHired ?d)
(not(assigned_employee_to_project ?d ?p))
(not(qualified_for_assigning_round_2 ?d ?p))))) )
```

(and

(project_for_round_2 ?p)))))

(:action **Assign-projects-round-2**

:parameters (?d - applicant ?t - title ?p - project)

:precondition (and

(employee_project_counts_initialized)

(isHired ?d)

(not(assigned_employee_to_project ?d ?p))

(applicant_for_title ?d ?t)

(>(titles_per_project ?t ?p) 0)

(<(number_of_projects_assigned_to_employee ?d) (/(total_per_title_from_each_project ?t)(number_of_titles_for_hiring ?t)))

(project_for_round_2 ?p)

)

:effect (and

(assigned_employee_to_project ?d ?p)

(increase(number_of_projects_assigned_to_employee ?d) 1)

(decrease(titles_per_project ?t ?p) 1)

))

(:action **Mark-completion-of-assigning**

:parameters ()

:precondition (and

(employee_project_counts_initialized)

(not(all-assigned))

(forall (?t - title) (not (exists (?d -applicant)(and(applicant_for_title ?d ?t)(isHired ?d)

(<(number_of_projects_assigned_to_employee ?d) (-(/(total_per_title_from_each_project ?t

)(number_of_titles_for_hiring ?t))1))))))

)

:effect (and

;when every project is assigned with required count of titles

(when (forall(?t - title ?p - project)(and(=(titles_per_project ?t ?p)0))) (all-assigned))

)))

Sample problem instance:

(define (problem P4) (:domain V6_software)

(:objects

a1 a2 a3 a4 a5 a6 a7 a8 - applicant

SE Manager - position

p1 p2 p3 - project

male female transgender - gender

Asian European-American-Non-Hispanic African-American American-Indian - race

straight gay - sexual-orientation

disabled none - disabilities

white black brown - color

)

(:init

(applicant-race a1 Asian)(applicant-so a1 straight)

(applicant-gender a1 male)(applicant-disabilities a1 none)

(applicant-color a1 white)(applicant-race a2 Asian)

(applicant-so a2 straight)(applicant-gender a2 male)

(applicant-disabilities a2 none)(applicant-color a2 white)

(applicant-race a3 Asian)(applicant-so a3 straight)

(applicant-gender a3 male)(applicant-disabilities a3 none)

(applicant-color a3 white)(applicant-race a4 African-American)

(applicant-so a4 gay)(applicant-gender a4 male)

(applicant-disabilities a4 none)(applicant-color a4 black)

(applicant-race a5 European-American-Non-Hispanic)

(applicant-so a5 straight)(applicant-gender a5 male)

(applicant-disabilities a5 none)(applicant-color a5 white)

(applicant-race a7 European-American-Non-Hispanic)

(applicant-so a7 straight)(applicant-gender a7 male)

(applicant-disabilities a7 none)(applicant-color a7 white)

(applicant-race a8 European-American-Non-Hispanic)

(applicant-so a8 straight)(applicant-gender a8 male)

(applicant-disabilities a8 none)(applicant-color a8 white)

(isWhite a8)(isWhite a7)(isWhite a2)(isWhite a1)(isWhite a3)

(isTransgender a8) (isEuropean-American-Non-Hispanic a8) (isEuropean-American-Non-Hispanic a7)

(isDisabled a5)

(=(applicants_count) 0)(=(projects_count) 0)

(=(hiring_diversity_count) 0)(=(projects_diversity_count) 0)

(=(positions_per_project SE p1) 4)(=(positions_per_project SE p2) 4)

(=(positions_per_project SE p3) 4)(=(positions_per_project Manager p1) 1)

(=(positions_per_project Manager p2) 1)(=(positions_per_project Manager p3) 1)

(=(Hired_count)0)

(=(minority_applicant_count) 0)(=(majority_applicant_count) 0)

(=(minority_applicants_hired) 0)(=(majority_applicants_hired) 0)

(=(number_of_openings_per_title SE) 6)(=(number_of_openings_per_title Manager) 2)

(=(minimum_experience_required_for_title SE) 3)(=(minimum_experience_required_for_title Manager) 10)

(=(number_of_positions_for_hiring SE) 6)(=(number_of_positions_for_hiring Manager) 2)

(=(total_per_position_from_each_project SE) 12)(=(total_per_position_from_each_project Manager) 3)

(=(applicant_experience a1) 2)(=(applicant_experience a2) 4)

(=(applicant_experience a3) 3)(=(applicant_experience a4) 3)

(=(applicant_experience a5) 5)(=(applicant_experience a6) 5)

(=(applicant_experience a7) 3)(=(applicant_experience a8) 6)

(applicant_for_title a1 SE)(hasMasters a1)

(applicant_for_title a2 SE)(hasBachelors a2)

(applicant_for_title a3 SE)(hasMasters a3)

(applicant_for_title a4 SE)(hasPhD a4)

(applicant_for_title a5 Manager)(hasPhD a5)

(applicant_for_title a6 SE)(hasMasters a6)

(applicant_for_title a7 SE)(hasBachelors a7)

(applicant_for_title a8 Manager)(hasPhD a8)

)

(:goal (and(all-assigned)))

)