APPENDIX B

Using SelfPlanner through its API—An Example

Let us define three activities, two located at the university, and one at the town center. One will be interruptible, one non-interruptible and the final one will be non-interruptible periodic. Completing the non-interruptible one before the interruptible one is a strong preference, but not a necessity.

Listing B.1: Using The SelfPlanner API Example in Java

```
// Initialize the data structures
Console console = new Console() {
    @Override
    public void write(String s) {
        System.out.println("SelfPlanner: " + s);
    }
};
int id = 100;
Date now = new Date();
long ts = now.getTime();
TaskManager tmgr = new TaskManager("" + id + "_" + ts);
LocationManager lmgr = new LocationManager("" + id + "_" + ts);
BinaryManager ordconstraints = new BinaryManager("" + id + "_" + ts, tmgr);
TemplateManager templates = new TemplateManager("" + id + "_" + ts);
DomainManager dmg = new DomainManager("" + id + "_" + ts);
LocationClassManager lcmgr = new LocationClassManager("" + id + "_" + ts);
BinaryManager dminconstraints = new BinaryManager("" + id + "_" + ts, tmgr);
BinaryManager dmaxconstraints = new BinaryManager("" + id + "_" + ts, tmgr);
BinaryManager iconstraints = new BinaryManager("" + id + "_" + ts, tmgr);
MyPlannerData data = new MyPlannerData("" + id + "_" + ts, tmgr, lmgr, ordconstraints,
    templates, dmg, lcmgr, dminconstraints, dmaxconstraints, iconstraints);
// Create the clear domain template
Template clearDomain = new Template(DomainAction.ActionAffects.DAY, "Clear Domain", -1);
clearDomain.setMultipleActions(0, DomainAction.Action.NOT_INC, 48);
// Define the problem instance
// Locations
```

```
Location univ = new Location(0, "University");
univ.setLatLng("40.62554905578553,22.960052490234375");
lmgr.addLocation(univ);
Location center = new Location(1,"Town Center");
center.setLatLng("40.626330777594895,22.9489803314209");
lmgr.addLocation(center);
// Get the real distance using the Google Distance Matrix API
LocationUtils.createLocationPairUsingDistanceMatrix(univ, center, lmgr);
try {
    Thread.sleep(30);
} catch (InterruptedException e) {}
LocationUtils.createLocationPairUsingDistanceMatrix(center, univ, lmgr);
// Enable Traveling Time Minimization
data.setLocationDistanceUtility(LOCATION_DISTANCE_UTILITY);
// Interruptible Activity
Domain d1 = new Domain(startDate, endDate);
d1.addTemplate(clearDomain, null, null, DomainAction.Action.NOT_INC);
for (Date[] interval : intervalsOfActivity1) {
    d1.addManualAction(new ManualAction(interval[0],
        interval[1], DomainAction.Action.INC));
}
DomainPrefs prefs1 = new NewDomainPrefs(NewDomainPrefs.NLPref.MORNING, TEMPORAL_UNARY_UTIL);
Task t1 = new Task(0, "Work on project", 150, 300, extraDurationUtil1, "University", d1,
    prefs1, null, activityUtility1, 1.0, 30, 90, 180, 1440);
tmgr.addTask(t1);
// Non-Interruptible Activity
Domain d2 = new Domain(startDate, endDate);
d2.addTemplate(clearDomain, null, null, DomainAction.Action.NOT_INC);
for (Date[] interval : intervalsOfActivity2) {
    d2.addManualAction(new ManualAction(interval[0],
        interval[1], DomainAction.Action.INC));
}
DomainPrefs prefs2 = new NewDomainPrefs(NewDomainPrefs.NLPref.AFTERNOON, TEMPORAL_UNARY_UTIL
    );
Task t2 = new Task(1, "Visit library", 120, 180, extraDurationUtil2, "Town Center", d1,
    prefs2, null, activityUtility2, 1.0);
tmgr.addTask(t2);
// Periodic Activity
Domain d3 = new Domain(startDate, endDate);
d3.addTemplate(clearDomain, null, null, DomainAction.Action.NOT_INC);
for (Date[] interval : intervalsOfActivity3) {
    d3.addManualAction(new ManualAction(interval[0],
        interval[1], DomainAction.Action.INC));
}
DomainPrefs prefs3 = new NewDomainPrefs(NewDomainPrefs.NLPref.AFTERNOON, TEMPORAL_UNARY_UTIL
    );
PeriodicPrefs.Period period = PeriodicPrefs.Period.DAILY;
PeriodicPrefs pprefs = new PeriodicPrefs(period, PeriodicPrefs.Range.END_BY_DEADLINE, true,
    true);
Task t3 = new Task(2, "Have lunch", 30, 60, extraDurationUtil1, "University", d1, prefs3,
    pprefs, activityUtility3, 1.0);
tmgr.addTask(t3);
BinaryPreference p1 = new BinaryPreference(1, 0, binaryUtil, BinaryPreference.ORD);
ordconstraints.addPreference(p1);
```

```
74 // Call SelfPlanner to solve the problem instance
75 SelfPlannerClient client =  new SelfPlannerClient("oaristotelis.uom.gr", console, PRIV_KEY,
       data, "-k 7000");
76 Solution[] solutions = null;
77 TaskSolved[] solved;
78 try {
79     solutions = client.simpleConnectAndSolve(NUM_OF_PLANS, univ); // The current location is
           at the university
80 } catch (TooManyClientsException ex) {
81     System.out.println("SelfPlanner overload. Exiting.");
82     System.exit(1);
83 }
84 if (solutions == null || solutions.length == 0)
85     System.out.println("No solution was returned");
86 else { // Print Plans
87     int pn,pn2;
88     for (int i=0;i<NUM_OF_PLANS;i++) {
89         solved = solutions[i].tasks();
90         pn = i + 1;
91         System.out.println("Printing plan " + pn);
92         for (TaskSolved element : solved) {
93             for (int j=0;j<element.solution().length;j++) {
94                 for (int k=0;k<element.solution()[j].length;k++) {
95                     pn = j + 1;
96                     pn2 = k + 1;
97                     System.out.println(element.task().name() + ":Period " + pn + "," + ",
                           part " + pn2);
98                     System.out.println(element.solution()[j][k].getStartTime() + " : " +
                           element.solution()[j][k].getEndTime());
99                 }
100             }
101         }
102     }
103 }
```