Universität Augsburg, LSt. Softwaretechnik, A. Schiendorfer, O. Kosak

Selbst-organisierende, adaptive Systeme (WS 16/17)

Übungsblatt 10 (Bearbeitung bis: 18.01.2017, 23:59 Uhr)

Task-Scheduling mit Mechanismus-Design

Aufgabe 1. ***(Vickrey-Clarke-Groves Mechanismus)***

Um einen besseren Eindruck für die Funktionsweise und Eigenschaften des Vickrey-Clarke-Groves Mechanismus zu erlangen, lösen Sie in dieser Übung ein einfaches Ressourcenallokationsproblem.

Definition 1: Vickrey-Clarke-Groves (VCG) Mechanismus

Der Vickrey-Clarke-Groves Mechanismus ist ein direkter quasilinearer Mechanismus (κ, p), mit

$$\kappa(\hat{v}) = \arg\max_{x \in X} \sum_i \hat{v}_i(x)$$

$$p_i(\hat{v}) = \sum_{j \neq i} \hat{v}_j\left(\kappa(\hat{v}_{-i})\right) - \sum_{j \neq i} \hat{v}_j(\kappa(\hat{v}))$$

Nehmen Sie an, ein neuer Berechnungsjob steht zur Verteilung. Drei Rechner A, B und C berechnen die geschätzte Transportzeit dieses Jobs in Minuten

- A: 5 Minuten
- B: 7 Minuten
- C: 12 Minuten

Das System vereinbart eine Entschädigung von 3 € pro Minute und bittet die Rechner um *wahrheitsgemäße* Einschätzung der Kosten, um den Job an den günstigsten Rechner zu vergeben. Einem Rechner entstehen auch tatsächliche Kosten, weshalb der Nutzen $v_i < 0$ ist, wenn i ausgewählt wird.

a) Definieren Sie die Auswahlmenge X und die wahren Bewertungen v_i für $i \in \{A, B, C\}$.

b) Wenden Sie VCG für die dominante Strategie $\hat{v}_i = v_i$ für alle Agenten an; welche Entscheidung wird getroffen und welche Bezahlungen ergeben sich daraus für die Agenten?

c) Was passiert, wenn ausgewählte Agenten ihre Bewertung nach oben oder unten verändern hinsichtlich der Auszahlungen?

Aufgabe 2. ***(Task-Scheduling)***

In dieser Übung implementieren Sie einfache Varianten von VCG und Kompensations- und Strafmechanismus für das Task-Scheduling-Problem aus der Vorlesung. Dadurch schaffen Sie eine Simulationsumgebung, in der Sie sich davon überzeugen, dass Kompensations- und Strafmechanismen anreizkompatibel bezgl. den minimalen Jobbearbeitungszeiten sind. Genauere Informationen zur Problembeschreibung finden Sie in [**?**, Kap. 10.6.1] (`http://www.masfoundations.org/download.html`).[1]
Basisdatentypen sind für Sie in der Datei `TaskSchedulingStartingPoint.zip` als Eclipse-Java-Projekt vorbereitet. Machen Sie sich zunächst mit den vorhandenen Klassen vertraut und lokalisieren Sie die formalen Definitionen im Programmcode. Passen Sie auch die vorgebenen Dateien nach Ihren Vorstellungen an.

[1] Achtung, die Clarke-Steuer wird mit falschem Vorzeichen angegeben, siehe `http://www.masfoundations.org/errata.html`

a) In `test/isse.LectureExample` finden Sie Testfälle für die Beispiele aus der Vorlesung (VL 10, Seite 29). Die Objektstruktur für dieses Beispiel wird angelegt und die entsprechenden Mechanismen zur Auswahl einer geeignenten `Shift` verwendet. Entwickeln Sie damit testgetrieben die Auswahlfunktionen der beiden Mechanismen. Sie können davon ausgehen, dass wir Problemgrößen betrachten, die Brute-Force Suche (z.B. Tiefensuche) über X zulassen. Welche Laufzeitkomplexität zeigt ihr Suchalgorithmus?

b) Erweitern Sie die Testfälle um die Bezahlungsfunktionen und implementieren Sie daraufhin die Bezahlungsfunktionen in den Mechanismen.

c) Implementieren Sie ein Experimentier-Setup, in welchem Sie für folgende Parameter ein zufälliges `SchedulingProblem` erzeugen können:

   - n Agenten
   - m Tasks
   - $0 < t_{\min} < t_{\max}$ die Schranken für die minimalen Bearbeitungszeiten $t_{i,j}$.

   Zufällig werden die (minimalen) Bearbeitungszeiten $t_{\min} \leq t_{i,j} \leq t_{\max}$ gewählt.

d) Implementieren Sie einen dedizierten `LyingAgent`, welcher bei seinen angegeben Zeiten lügt, allerdings bei den tatsächlichen Zeiten immer *zumindest* die Minimalzeit benötigt. Machen Sie den Lügenmodus ausschaltbar (also als Faktor) und vergleichen Sie die Bezahlungen, die ihr Lügner bekommt, wenn er von seinen Minimalzeiten abweicht sowohl für:

   - Angegebene Zeiten
   - Tatsächlich eingetretene Zeiten (beachten Sie, dass diese nie kürzer als die minimalen Zeiten sein dürfen).

   Führen Sie ein Experiment aus, in welchem Sie den privaten Nutzen des Lügners mit einem wahrheitsgemäßen Agenten auf Signifikanz vergleichen (T-Test). Achten Sie auf eine saubere Ausarbeitung im Sinne guter Softwaretechnik-Prinzipien (Kapselung, Wiederverwendbarkeit, Testabdeckung).