Mechanism Design*

Christoph Hermann

22. Mai 2006

Inhaltsverzeichnis

*(Draft) Work in progress...

1 Einführung in Mechanism Design

Verschiedene Bereiche der Forschung in der Informatik beschäftigt sich mit Protokollen, Algorithmen und Datenstrukturen für Computernetzwerke. Die Entwickler von verteilten Algorithmen bzw. Algorithmen mit mehreren Teilnehmern (Agenten) gehen fast immer davon aus, dass die sich im Netzwerk befindlichen Agenten sich wie vorgesehen verhalten, zumindest wenn man von solchen Teilnehmern absieht welche sich absichtlich falsch oder bösartig verhalten um anderen zu schaden.

Ein faires (bzw. sinnvolles) Verhalten aller Teilnehmer kann aber nicht immer angenommen werden. In einem nicht zentral kontrollierten Netzwerk wie dem Internet haben verschiedene Besitzer unterschiedliche Interessen und verfolgen unterschiedliche Ziele die nicht zwingend mit einem (*dem*) „globalen" Ziel übereinstimmt.

Um die Interessen der einzelnen Teilnehmer mit einem globalen Ziel in Einklang zu bringen wurde aus den in der Spieltheorie und der Informatik gewonnenen Kenntnissen ein neues Forschungsgebiet „Mechanism Design" entwickelt, welches sich genau dieser Problemstellung annimmt. In der wirtschaftswissenschaftlichen Anwendung der Spieltheorie wurden bisher Anreize/Anreizsysteme als primärer Fokus der Forschung gesehen, während in der Informatik der Fokus auf der Komplexität/Berechenbarkeit der Probleme lag. Anreize (u.a. Auszahlungen) werden in der Spieltheorie verwendet um einen Agenten dazu zu bringen, sich nach bestimmten Regeln zu verhalten, da er so (unter Beachtung der Regeln) seinen Nutzen (Auszahlung/Wert) maximieren kann.

2 Definitionen

Bevor wir vertiefend in das Themengebiet des Mechanism Design einsteigen ist ein grundlegendes Verständnis der Spieltheorie notwendig. Dazu werden im Folgenden ein paar

Beispiele gegeben und die grundlegenden Begriffe und Definitionen erläutert.

2.1 Spieltheorie

Grundlegende Definitionen

N	Anzahl Agenten (Spieler)
S_i	Strategiemenge des Agenten i
s_i	Gewählte Strategie des Agenten i
p_i	Auszahlung des Agenten i

- Gleichzeitige Auswahl der jeweiligen Strategie: Jeder Agent i wählt eine Strategie $s_i \in S_i$
- Niemand kann den Spielzug eines Gegners beobachten
- Die Strategiekombination $(s_1, s_2, \ldots, s_n)$ führt zu der Auszahlung $(p_1, p_2, \ldots, p_n)$ für die N Agenten
- All diese Informationen sind allen Agenten bekannt
- Ein *Gleichgewicht* $s^* = (s_1^*, s_2^*, \ldots, s_n^*)$ ist eine Strategiekombination welche aus der *besten* Strategie für jeden der N Agenten im Spiel besteht.
- *Gleichgewichtsstrategien* sind die Strategien welche Agenten wählen um ihren eigenen Nutzen zu maximieren, unter der Bedingung dass die anderen Agenten sich genauso verhalten.

Was ist nun die *beste* Strategie? Die Auswahl der besten Strategie ist sehr stark vom Spiel abhängig.

Paradoxon: Gefangenendilemma Das Gefangenendilemma zeigt wie individuell rationale Entscheidungen zu einem kollektiv schlechterem Ergebnis führen können.

Zwei Gefangene werden beschuldigt eine Straftat begangen zu haben. Den Gefangenen wird nun ein Handel vorgeschlagen der beiden bekannt ist: Sie haben die Wahl entweder zu schweigen und Ihre derzeit nachweisbare Strafe abzusitzen (1 Jahr), oder den Partner zu verraten, der dann die volle Strafe abbekommt (5 Jahre aufgrund der Zeugenaussage) und selbst entlassen zu werden. Verraten beide den anderen, so erwartet Sie jeweils eine Strafe von 4 Jahren. Die Gefangenen haben keinerlei Möglichkeit sich untereinander abzusprechen.

Daraus ergibt sich dann folgende Entscheidungsmatrix:

Gefangener A \ Gefangener B	B verrät A	B schweigt
A verrät B	A: -4 / B: -4	A: 0 / B:-5
A schweigt	A: -5 / B: 0	A: -1 / B: -1

Die existierenden Ausgänge des Verfahrens hängen also nicht nur von der eigenen Entscheidung ab, sondern auch von der Entscheidung des Komplizen. Betrachtet man die individuelle Entscheidung der beiden Gefangenen, so erscheint es vorteilhaft den anderen zu verraten, da man so das eigene Strafmaß mildert (frei kommt). Der Gefangene überlegt sich, dass wenn sein Komplize gesteht, so reduziert sich seine Strafe von 5 auf 4 Jahre Gefängnis. Falls er sogar schweigt, so kann er seine Strafe von 1 Jahr auf 0 Jahre reduzieren. Diese individuelle rationale Entscheidung hängt nicht vom Verhalten des anderen ab, es erscheint also vorteilhaft immer zu gestehen. Eine solche Strategie die unabhängig von den Strategien der Mitspielern ist bezeichnet man als *dominant*.

Da nun aber beide Gefangenen diese rationale Entscheidung fällen, stellen sie sich insgesamt schlechter (beide bekommen 4 Jahre aufgebrummt) als wenn Sie beide schweigen würden (jeweils 1 Jahr).

Definition 2.1. Dominante Strategie

Eine *dominante Strategie* ist eine Strategie, die immer besser ist, als alle anderen Strategien des Agenten, egal was die anderen Agenten für eine Strategie wählen.

Eine Strategie s_i^* von Spieler i dominiert alle anderen seiner Strategien s_i (streng oder strikt), falls sie unabhängig von der Strategiewahl s_{-i} der anderen Spieler immer strikt die höchste Auszahlung generiert, d.h. $u_i(s_i^*, s_{-i}) > u_i(s_i, s_{-i})$, für alle $s_i \neq s_i^*$ und alle s_{-i}. Eine solche Strategie wird (streng oder strikt) dominante Strategie genannt.

Kampf der Geschlechter Bei diesem bekannten Spiel geht es um die Auswahl des Abendprogramms für einen gemeinsamen Abend von Mann und Frau. Zur Auswahl stehen der Besuch eines Konzerts oder eines Fußballspiels. Das Fußballspiel wird vom Mann präferiert, das Konzert von der Frau. Beide müssen unabhängig voneinander Ihre Wahl treffen.

Die Entscheidungsmatrix sieht bei diesem Spiel folgendermaßen aus:

Mann \ Frau	Fußball	Konzert
Fußball	M: 2 / F: 1	M: 0 / F: 0
Konzert	M: 0 / F: 0	M: 1 / F: 2

Da die beiden den Abend gemeinsam verbringen wollen, nützt es Ihnen nichts, wenn jeder von Ihnen der präferierten Beschäftigung nachgeht. Geht die Frau also zum Konzert, so ist für den Mann die beste Entscheidung auch dorthin zu gehen. Geht der Mann zum Fußball, so ist es für die Frau das beste mitzugehen.

Das Problem dieses Spiels ist nun, dass es hier keine dominante Strategien gibt. Gehen beide Partner ihrer Lieblingsbeschäftigung nach, so treffen Sie sich nicht. Denken sie aber beide, dass es besser wäre zumindest zusammen zu sein und geben beide nach, so treffen Sie sich wieder nicht. Es gibt für Mann und Frau keine optimale Strategie die unabhängig von der Entscheidung des Partners ist.

Als Ausweg können die beiden Spieler zufällig wählen (die Entscheidung randomisieren) wo Sie den Abend verbringen werden.

Definition 2.2. Pareto-Effizienz
Ein Strategie-Set ist pareto effizient, wenn es kein alternatives Set gibt, die jeden Agenten zumindest gleich gut gestellt lässt und mindestens einen Agenten besser stellt.

Nash Gleichgewicht

Definition 2.3. Nash Gleichgewicht
Das *Nash-Gleichgewicht* beschreibt in Spielen einen Zustand des strategischen Gleichgewichts in dem kein Spieler seine Situation verbessern kann in dem nur er alleine seine Strategie verändert.

Im Gefangenendilemma ist die Situation in der beide Gefangenen Ihren Komplizen verraten haben ein *pareto-ineffizientes Nash-Gleichgewicht*.

2.2 Vickrey Auktion

Prinzip der Vickrey Auktion Bei der Vickrey-Auktion erhält der höchst bietende Bieter den Zuschlag genau wie bei einer einfachen Auktion, jedoch muss er nicht sein volles Gebot für den ersteigerten Artikel bezahlen, sondern lediglich den Preis des Gebots den der zweit höchste Bieter geboten hat.

Das Interessante an der Vickrey-Auktion ist, dass es für den Bieter vorteilhaft ist, ein Gebot in der Höhe seiner Wertschätzung für den Artikel abzugeben im Gegenteil zur Erstpreisauktion (wo es vorteilhaft wäre niedriger zu bieten als die eigene Wertschätzung um einen Gewinn zu erwirtschaften).

Die Abgabe der Gebote erfolgt bei einer Vickrey-Auktion geheim und die Gebote werden am Ende der Auktion auch nicht aufgedeckt. Man erfährt lediglich den Gewinner der Auktion und den bezahlten Preis.

Bsp. einer Vickrey Auktion Gegeben einen ganz einfachen Fall mit zwei Bietern A und B die auf einen Artikel X bieten (wir gehen davon aus, dass der Auktionator die Gebote nicht manipuliert).

- tatsächlicher Wert des Artikels X für A: 32 GE[1]

Für A ergeben sich nun 3 Mögliche Strategien:

1. Überbieten
   Im Falle eines **Überbietens** (falls A sowieso der Gewinner wäre) gewinnt A nichts (der Artikel ist ihm ja maximal 32GE wert) **(Fall a)**. Wäre A nicht der eigentliche Gewinner (jemand anders hat eine höhere Wertschätzung für den Artikel), so bezahlt er unter Umständen mehr als ihm der Artikel wert ist **(Fall b)**.

   a) A bietet 34 GE
      B bietet 30 GE
      => A gewinnt und zahlt 30 GE.

[1] GE = Geldeinheiten

b) A bietet 34 GE
B bietet 33 GE
=> A gewinnt und zahlt 33 GE, d.h. er macht **1 GE Verlust**.

2. Unterbieten
Im Falle eines **Unterbietens** (falls A sowieso der Gewinner wäre) gewinnt A nichts (er zahlt ja sowieso nur das 2.höchste Gebot) **(Fall a)**. Allerdings geht er das Risiko ein die Auktion zu verlieren, da ein anderer Bieter eine geringfügig höhere Wertschätzung haben bzw. ein höheres Gebot abgeben könnte.

   a) A bietet 30 GE
   B bietet 29 GE
   => A gewinnt und zahlt sowieso nur 29GE

   b) A bietet 28 GE
   B bietet 29 GE
   => A **verliert** und bekommt nichts. Hätte A 32 GE geboten, hätte er den Zuschlag erhalten und nur 29 zahlen müssen.

3. Den genauen Wert bieten
Bei der Vickrey Auktion ist es immer die beste Strategie den genauen Wert zu bieten. Die dominante Strategie ist es also immer die Wahrheit zu sagen (Gebot in Höhe der Wertschätzung) unabhängig davon was die anderen Bieter tun.

2.3 Mechanism Design

Spieltheorie ist eigentlich die Erforschung des Verhaltens von unabhängig eigennützig agierenden Agenten. Mechanism Design versucht Methoden zu entwickeln mit denen man Systeme/Algorithmen entwerfen kann, so dass das Verhalten der einzelnen Agenten in einem gewünschten Verhalten zur Erreichung eines globalen Ziels (Lösung eines bestimmten Problems) resultiert. Die „Mechanismen" (Verhaltensregeln) sind Angaben zu Nutzen und Auszahlungen der Agenten, welche den Agenten Anreize liefern sich so zu verhalten, dass das globale Ziel erreicht wird.

2.3.1 Definitionen

- N Agenten, jeder Agent besitzt *private* Informationen; den *Typ*, $t_i \in T_i$(Set von möglichen Typen)
- Agent i kennt nur seinen eigenen Typ, aber nicht den der anderen Agenten
- Die anderen Agenten wissen, dass das Set der möglichen Typen für Agent i T_i ist.
  - Bsp. bei einer Auktion: $T_i = [75, 100]$: Die Agenten schätzen den Wert eines Guts zwischen 75 und 100 ein (dies ist allen Agenten bekannt). $t_i = 80$: Die Wertschätzung des Guts eines Agenten i (dieser Wert ist den anderen Agenten nicht bekannt).

- O ist die Menge der möglichen Ergebnisse
- Ergebnisspezifikation g: Für ein gegebenes Set an Typen $(t_1, t_2, \ldots, t_n)$ gibt g ein gültiges Ergebnis o an.
  - Bsp. bei einer Auktion: O sind die verschiedenen möglichen Gewinner der Auktion; $g : max(t_1, \ldots, t_n)$ (Ermittle den Gewinner der Auktion durch das höchste Gebot.
- Wenn o das Ergebnis ist, $t_i$der Typ, dann ist Agent i's Nutzen gegeben durch die Funktion $v_i = (o, t_i)$.
  - Bsp. bei einer Auktion: Wenn Agent i die Auktion gewinnt, dann ist sein Ergebnis gleich dem Wert des Guts, sonst ist es 0.
- Wenn p_i die (Aus-)Zahlung des Agenten i ist, dann ist der Nutzen des Agenten i: $u_i = v_i(o, t_i) + p_i$
  - Bsp. bei einer Auktion: Wenn die Wertschätzung des Agenten für das Gut 100 ist, und er 90 bezahlt (Auszahlung: -90), dann ist sein Gewinn $100 + (-90) = 10$.
- Ziel des Agenten ist es also den erwarteten Nutzen zu maximieren.

2.3.2 Direkte und indirekte Mechanismen

Definition eines direkten Mechanismus

Definition 2.4. Direkte Mechanismen
Direkte Mechanismen sind Mechanismen bei denen die Agenten Ihren Typ t_i direkt angeben.

- Mechanismus: $M =< O, P >$
- T_i: Set der verschiedenen Typen eines Agenten i, $T = \prod_i T_i$
- Spezifikation des Ergebnisses: $g : T \rightarrow O$
- Wertschätzung: $v_i(o, t_i)$; Nutzenfunktion: $u_i = v_i(o, t_i) + p_i$
- Der Mechanismus gibt an: Ergebnis o als Funktion der Typen der Agenten, sowie Auszahlung p als Funktion der Typkombination

Definition eines indirekten Mechanismus:

Definition 2.5. Indirekte Mechanismen
Indirekte Mechanismen sind Mechanismen bei denen Agenten Ihren Typ t_i nicht direkt angeben, sondern den Wert einer Funktion in Abhängigkeit Ihres Typs (Strategie eines Agenten i).

- Mechanismus: $M =< S, O, P >$
- T_i: Set der verschiedenen Typen eines Agenten i, $T = \prod_i T_i$
- $S = \prod_i S_i$ mit S_i Strategie des Agenten i (eine Strategie ist eine Funktion des Typs des Agenten i)
- Spezifikation des Ergebnisses: $g : S \rightarrow O$
- Wertschätzung: $v_i(o, t_i)$; Nutzenfunktion: $u_i = v_i(o, t_i) + p_i$
- Der Mechanismus gibt an: Ergebnis o als Funktion der Strategiekombination der Agenten, sowie Auszahlung p als Funktion der Strategiekombination

Definition 2.6. Utilitaristische Mechanismen
Ein Mechanismus ist *utilitaristisch*, wenn es das Ziel des Mechanismus ist, den Wert des Nutzen zu maximieren: $max_o \sum_i v_i(o, t_i)$

2.3.3 Strategiebeständigkeit

Definition 2.7. Strategiebeständigkeit (strategy proof/truthful)
Wenn wahrheitsgemäße Berichterstattung eine dominante Strategie innerhalb eines Mechanismus ist, so wird der Mechanismus als *strategiebeständig (strategy proof/truthful)* bezeichnet.

Ein Agent gibt also seine wahre Nutzenfunktion an, anstatt diese strategisch zu verändern.

2.3.4 Group-Strategy proof

Definition 2.8. Gruppenstrategiebeständigkeit
Ein Mechanismus ist *gruppenstrategiebeständig*, wenn keine Gruppe von Agenten kooperativ einen Vorteil gewinnen können wenn sie falsch (unwahrheitsgemäß) über Ihren privaten Typ berichten.

Formal: Sei $t = (t_1, \ldots, t_n)$ ein Typvektor und $S \subseteq \{1, \ldots, n\}$ eine kooperativ strategisch agierende Gruppe von Agenten. Sei $a_s = \{a_i\}_{i \in S}$ die Gruppenstrategie von S, d.h. für $i \in S$, $a_i \neq t_i$. Sei $t_{\overline{S}} = \{t_i\}_{i \notin S}$ die Menge der Typen der Agenten die nicht in der strategischen Gruppe sind. Dann ist $o' = o(t_{\overline{S}}, a_s)$ und $q_i = p_i(t_{\overline{S}}, a_s)$ das Ergebnis und die Zahlungen die der Mechanismus anhand der Angaben $t_{\overline{S}}$ und a_s berechnet, wenn jeder der Agenten $i \in S$ den Typ a_i angibt und jeder Agent $i \notin S$ den wahren Typ t_i

angibt. Seien $o = o(t)$ und $p_i = p_i(t)$ die Ergebnisse und Zahlungen wenn alle Agenten wahrheitsgemäß Ihren Typ angeben würden. Der Mechanismus ist nun gruppenstrategiebeständig, wenn für alle t, S, und a_S entweder $u_i(o', t_i) = u_i(o, t_i)$ für alle $i \in S$, oder es gibt zumindest einen Agent $i \in S$, für den gilt: $u_i(o', t_i) < u_i(o, t_i)$. Das bedeutet, wenn die Gruppenstrategie a_S ein besseres Ergebnis für einen Agenten $i \in S$ liefert, so muss zumindest ein anderer Agent $j \in S$ ein schlechteres Ergebnis erzielen.

2.3.5 Vickrey-Clarke-Groves pricing mechanisms

Der Vickrey-Clarke-Groves Mechanismus wurde von W. Vickrey, E. H. Clarke, und T. Groves entwickelt und gilt als einer der wichtigsten Mechanismen schlechthin.

- Der (direkte) Mechanismus ist utilitaristisch: $o(t) \in max_p \sum_i v_i(o, t_i)$
- Nutzen: $u_i = v_i(o, t_i) + p_i$
- Dann sichert folgende Auszahlungsfunktion, dass der Mechanismus strategiebeständig (Wahrheitsgemäße Berichterstattung ist dominante Strategie) ist: $p_i(t) = \sum_{j \neq i} v_j(o, t_j) + h_i(t_{\bar{i}})$
- h_i ist hier eine (nicht näher definierte) Funktion der Typen t der anderen Agenten

Satz 2.1. *(Groves (1973))*
Ein VCG-Mechanismus ist strategiebeständig.

Aus diesem Grund bietet der VCG-Mechanismus eine Lösung für jedes utilitaristische Problem (wenn man davon absieht, dass es eventuell andere dominante Strategien gibt als die wahrheitsgemäße Berichterstattung). Es ist bekannt (unter bestimmten Voraussetzungen), dass VCG-Mechanismen die einzigen strategiebeständigen Mechanismen für utilitaristische Probleme sind [5].

Definition 2.9. Budgetbalanciertheit
Strikte Budgetbalanciertheit: Die Summe der gesamten Zahlungen der Agenten ist genau 0, das bedeutet keine Zahlungen werden in das System hineingebracht und keine Zahlungen fließen aus dem System hinaus: $\sum_{i=1}^{N} p_i = 0$.
Schwache Budgetbalanciertheit: Die Summe der gesamten Auszahlungen ist nicht negativ (der Mechanismus macht keinen Verlust): $\sum_{i=1}^{N} p_i >= 0$.

Definition 2.10. Effizienz
Ein Mechanismus ist *effizient*, wenn er eine utilitaristische Nutzenfunktion implementiert.

2.3.6 Impossibility Result

Satz 2.2. *Kein Mechanismus kann gleichzeitig* effizient, strategiebeständig *und* budgetbalanciert *sein.*

- Man *muss* eines der obigen Kriterien einschränken.
- Budget-balanciertheit ist ein notwendiges Kriterium für jedes „echte" System (niemand möchte Verlust machen).
- Effizienz ist auch notwendig, da kein Agent ineffiziente Ergebnisse akzeptieren würde.

Die Lösung für dieses Problem könnten Bayessche Nash Mechanismen sein.

2.3.7 Bayesscher Nash Mechanismus

Definition 2.11. Bayessches Spiel
Ein Bayessches Spiel ein Spiel in dem die Informationen über die Eigenschaften der anderen Agenten unvollständig sind. Das bedeutet dass in einem Bayesschen Spiel mindestens ein Agent sich nicht über den Typ der anderen Agenten im Klaren ist.

In einem Nicht-Bayesschen Spiel ist eine Strategieset ein Nash Gleichgewicht, wenn jede Strategie aus diesem Set eine optimale „Antwort" auf jede andere Strategie aus diesem Set darstellt. D.h. es existiert keine andere Strategie welche einen größeren Nutzen erzielen könnte, gegeben die Strategien der anderen Agenten.

Definition 2.12. Bayessches Nash Gleichgewicht
Ein Bayessches Nash Gleichgewicht ist definiert als eine Strategiekombination *und* Vermutungen über den Typ aller anderen Agenten welche den erwarteten Nutzen aller Spieler maximiert, gegeben die Vermutung über den Typ der anderen Agenten und gegeben die Strategien der anderen Agenten.

In einem Bayesschen Nash Mechanismus muss ein Bayessches Nash Gleichgewicht existieren. Der Mechanismus ist budget-balanciert und effizient. Das Problem hierbei ist es, ein Bayessches Nash Gleichgewicht zu finden (dies ist u.a. Gegenstand der aktuellen Forschung[2]).

3 Anwendungsgebiete des Mechanism Design

3.1 Shortest Path

Problemstellung Es existiert ein Kommunikationsnetzwerk modelliert durch einen gerichteten Graphen G, und zwei spezielle Knoten X und Y. Jede Kante des Graphen ist ein Agent. Jeder Agent e, besitzt eine private Information (seinen Typ) $t_e \geq 0$ welcher die Kosten des Agenten darstellt um eine einzelne Nachricht über diese Kante zu verschicken. Das Ziel ist es den kürzesten (preiswertesten) Pfad von X nach Y zu finden (d.h. eine Nachricht von X nach Y zu senden). D.h. die möglichen Ergebnisse des Problems sind die möglichen Pfade von X nach Y und das Ziel ist es die gesamten Kosten des Pfades zu minimieren. Der Nutzen eines Agenten e ist 0, wenn seine Kante nicht Teil des gewählten Pfades ist und $-t_e$ falls sie es ist.

[2] Es gibt bisher noch keine entscheidenden algorithmischen Vorgehensweisen für Bayessche Nash Mechanismen.

Strategiebeständige Implementierung Der folgende Mechanismus sichert, dass die dominante Strategie für jeden Agenten darin besteht, ihren wahren Typ t_e an den Mechanismus zu berichten (d.h. wenn alle Agenten wahrheitsgemäß berichten wird der kürzeste Pfad gewählt):

- Das Ergebnis wird bestimmt durch eine einfache kürzeste-Pfade-Berechnung.
- Die Zahlung p_e an den Agenten e ist 0, wenn e nicht auf dem kürzesten Pfad liegt und $p_e = d_{G|e=\infty} - d_{G|e=0}$ wenn e auf dem Pfad liegt.
- $d_{G|e=\infty}$ist hier die Länge des kürzesten Pfades der e nicht enthält (gemäß den berichteten Angaben der Agenten).
- $d_{G|e=0}$ ist die Länge des kürzesten Pfades unter der Annahme, dass die Kosten für e $t_e = 0$ sind (wieder gemäß den Angaben der Agenten).
- Dieser Mechanismus ist ein VCG Mechanismus: $d_{G|e=\infty}$entspricht $h_i(t_{-i})$ und $d_{G|e=0}$ entspricht $\sum_{j \neq i} v_j(t_j, o(t))$.

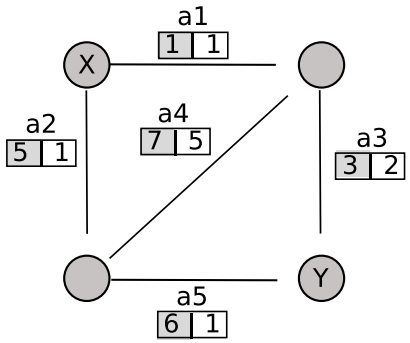

Abbildung 1: Beispiel eines ungerichteten Graphen

Beispiel

Berechnung der kürzesten Pfade Man sieht hier, dass eine unwahrheitsgemäße Berichterstattung zu einem negativen Ergebnis führt. Aus diesem Grund haben die Agenten keinerlei Anreiz von der wahrheitsgemäßen Berichterstattung abzuweichen.

Berechnung des kürzesten Pfades im Netzwerk (Komplexität) Die Frage die sich nun stellt ist: Wie kann man den kürzesten Pfad einfach (mit geringer Komplexität) berechnen? Zur Lösung dieses Problems schlagen John Hershberger und Subhas Suri folgenden Lösungsansatz vor:

- Sei $pfad(x, y) = (v_1, \ldots, v_k)$ der kürzeste Pfad zwischen x und y mit $v_1 = x$ und $v_k = y$.

a_1	a_3			$\sum$
1	3		=	*4*
a_1	a_4	a_5		$\sum$
1	7	6	=	14
a_2	a_4	a_3		$\sum$
5	7	3	=	15
a_2	a_5			$\sum$
5	6		=	11

Zahlung	Kosten bei Nutzung	Nutzen
$p_1 = 11 - 3 = 8$	1	7
$p_2 = 0$	5	0
$p_3 = 11 - 1 = 10$	3	7
$p_4 = 0$	7	0
$p_5 = 0$	6	0

Tabelle 1: bei wahrheitsgemäßer Berichterstattung (grau hinterlegt)

a_1	a_3			$\sum$
1	2		=	3
a_1	a_4	a_5		$\sum$
1	5	1	=	7
a_2	a_4	a_3		$\sum$
1	5	2	=	8
a_2	a_5			$\sum$
1	1		=	*2*

Zahlung	Kosten bei Nutzung	Nutzen
$p_1 = 0$	1	0
$p_2 = 3 - 1 = 2$	5	*-3*
$p_3 = 0$	3	0
$p_4 = 0$	7	0
$p_5 = 3 - 1 = 2$	6	*-4*

Tabelle 2: bei unwahrheitsgemäßer Berichterstattung (weiß)

- $d(x, y, G)$ ist die Distanz des kürzesten Pfades $pfad(x, y, G)$ von x nach y im Graphen G.
- Berechne für jedes $i \in \{1, \ldots k-1\}$ die Länge des kürzesten Pfades von X nach Y, der nicht die Kante $e_i = (v_i, v_{i+1})$ enthält.
- Man analysiert die kürzesten Pfade von X nach Y anhand der Kanten die einen Schnitt kreuzen. Dieses Set von Kanten welche den Schnitt kreuzen bezeichnen wir als $E_{cut} = E(V_x, V_y)$. Für jede Kante $(u, v) \in E_{cut}$ gilt $u \in V_x$ und $v \in V_y$.
- Für jede Kante $e_i = (v_i, v_{i+1})$ und einen Schnitt (V_x, V_y) (eventuell abhängig von e_i) berechne: $d(x, y; G \backslash e_i) = min_{\substack{(u,v) \in E_{cut} \\ (u,v) \neq e_i}} (d(x, u; G \backslash e_i) + c(u, v) + d(v, y; G \backslash e_i))$
  ($c()$ entspricht dem Typ des Agenten dieser Kante) (Man begeht jeden Pfad der eine alternative Kante des Schnitts enthält um von X nach Y zu gelangen.)

Betrachten wir den einfachen Fall (siehe Abbildung 2) in welchem $pfad(X, Y)$ alle Knoten des Graphen enthält. Wir definieren $cut(V_x, V_y)$ indem wir die Kante $e_i = (v_i, v_{i+1})$ entfernen. V_x ist also $V_x = \{v_1, \ldots, v_i\}$und $V_y = \{v_{i+1}, \ldots, v_n\}$. Zur Vereinfachung bezeichnen wir $E_i = E(V_x, V_y)$ ($E_i = E \backslash E_{cut}$). Für eine gegebene Kante $e_i = (v_i, v_{i+1})$ die entfernt werden soll, betrachten wir eine Schnittkante $(u, v) \in E_i$. Weil $pfad(x, u)$ in V_x enthalten ist, ist $d(x, u) = d(x, u, G \backslash e_i)$. Ebenso für $d(v, y)$. Diese Distanzen sind einfach die Längen der Teilpfade von $pfad(x, y)$ die jeden Punkt mit x bzw. y verbinden.

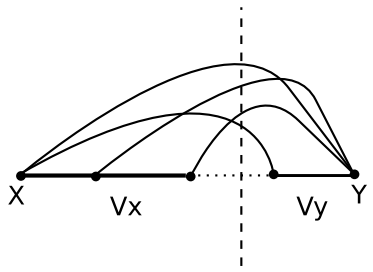

Abbildung 2: Die Kanten von E_i schneiden die vertikale gestrichelte Linie (Schnitt)

In diesem gegebenen Graphen vereinfacht sich also die zu berechnende Gleichung zu:

$$d(x, y; G \backslash e_i) = min_{\substack{(u,v) \in E_i \\ (u,v) \neq e_i}} d(x, u) + c(u, v) + d(v, y)$$

.

Um jetzt den kürzesten x-y-Pfad der eine Kante des Pfades $pfad(X, Y)$ weg lässt effizient zu berechnen wird obige Gleichung für jede $e_i = (v_i, v_{i+1})$ in der Sequenz von $i = 1$ bis $n - 1$ berechnet. Für ein gegebenes i minimieren wir $d(x, u) + c(u, v) + d(v, y)$ über alle $(u, v) \in E_i \backslash e_i$.

Mit einer naiven Vorgehensweise zur Berechnung der Auszahlungen der Agenten erreicht man so eine Komplexität von $O(n^2 \log n + nm)$ für Netzwerke mit nicht-negativen Kosten.

Der Trick ist jetzt E_{i+1} aus E_i zu berechnen. Der Unterschied zwischen E_i und E_{i+1}besteht aus genau den Kanten mit einem Endpunkt in v_{i+1}. Um E_{i+1} aus E_i zu berechnen fügen wir E_i genau die Kanten hinzu, deren linker Endpunkt v_{i+1} ist und wir entfernen von E_i genau die Kanten deren rechter Endpunkt v_{i+1} ist.

Dies lässt sich mit folgendem Algorithmus realisieren:

Algorithm 1 Pfad Algorithmus

1. Seien $left(e)$ und $right(e)$ die Indices der Endpunkte der Kante e. Seien L und R n-Elemente Arrays die Mengen von Kanten enthalten können. (Initialisiere diese mit der leeren Menge.) Sei Q eine Priority Queue mit (Gewichtung,Kante)-Paaren, indiziert nach der Gewichtung, leer initialisiert.

2. For each $e \in E \backslash pfad(X, Y)$
   If $left(e) < right(e)$, füge e in $L[left(e)]$ und $R[right(e)]$.

3. For $i = 0$ to $n - 1$

   a) For each $e = (u, v) \in L[i]$
   Füge (w, e) in Q ein mit Gewichtung $w = d(x, u; G \backslash e_i) + c(u, v) + d(v, y; G \backslash e_i)$.

   b) Entferne von Q alle (w, e) Paare mit $e \in R[i]$.

   c) Liefere die minimale Gewichtung in Q als den x-y-Abstand in G ohne e_i.

Im Schritt i enthält Q die Kanten von $E_i \backslash \{e_i\}$ und damit ist der x-y-Abstand ohne e_i korrekt berechnet. Nur die Operationen der Priority Queue benötigen nicht-konstante Zeit. Eine einfache Implementierung einer Priority Queue liefert eine Komplexität von $O(m \log m)$. Diese Komplexität kann man durch die Verwendung eines Fibonnaci-Heaps noch weiter verringern (n Anzahl der Knoten; m Anzahl der Kanten). Durch die Verwendung eines Fibonnaci-Heaps erreicht man eine Komplexität von $O(n \log n + m)$, dies entspricht der Komplexität bei der Berechnung durch Dijkstra (also keine Verschlechterung der Komplexität durch den Mechanismus).

3.2 Interdomain Routing

Das Internet ist unterteilt in mehrere separate administrative Domänen bzw. autonome Systeme (AS). Das Routing von Paketen geschieht auf zwei verschiedenen Ebenen:

- Intradomain Routing: Route innerhalb eines AS
- Interdomain Routing: Route zwischen verschiedenen AS: Hier wird derzeit das Border Gateway Protocol (BGP) genutzt.

BGP

- wird benutzt um Routing Informationen zwischen verschiedenen AS zu transportieren.
- leitet Informationen über die Topologie der AS weiter
- *Path Vector Protocol*
  - Knoten i speichert, für jedes AS j, den kürzesten AS-Pfad von i nach j
  - *kurz* im Sinne von Anzahl hops
  - Jeder Router sendet seine Routing-Tabelle an seine Nachbarn, die wiederum Ihre eigene Routing-Tabelle anhand dieser Informationen berechnen
  - Der Austausch von Routing-Tabellen findet statt sobald eine Änderung festgestellt wurde

BGP Modell

- Jedes AS wird als Knoten interpretiert
- N sei ein Set von Knoten (AS), $n =\| N \|$
- L sei ein Set von bidirektionalen Links in N, mindestens ein Link zwischen zwei Knoten
- T_{ij} Intensität des Datenverkehrs kommend von i in Richtung j
- c_k Weiterleitungskosten pro Paket des Knotens k
  - Weiterleitungsdaten sind Daten die weder von diesem AS kommt, noch an dieses AS gerichtet ist
  - AS haben keinerlei eigenes Interesse an der Weiterleitung von Daten
  - Zum Ausgleich wird jedes AS für die Weiterleitung bezahlt

Ziel:

- Minimierung der Kosten aller gerouteten Pakete (utilitaristisch)
- AS welche keinerlei Pakete weiterleiten zahlen nichts
- Benutzung von BGP als Basis zur Lösung dieses Problems
  - Erweiterungen eines bereits existierenden Protokolls sind weitaus einfacher zu entwickeln, als komplett neue zu entwerfen
- Es gibt keine zentrale Instanz im BGP Protokoll: Jeder Knoten berechnet seine Routing Tabellen und sendet sie an seine Nachbarn
- Veränderungen des BGP Protokolls sollten keinerlei neue Ansprüche an die Router bzgl. Berechnung und Kommunikation stellen

Strategiebeständiger Zahlungsmechanismus für Routing

- Jeder Agent k bekommt eine Zahlung für die Weiterleitung von Datenpaketen von i nach j = c_k(0 wenn die Weiterleitung keinerlei Kosten verursacht) + [Weiterleitungskosten des kürzesten Pfads ohne k - Weiterleitungskosten des kürzesten Pfades mit k]
- Einzigartiger strategiebeständiger Zahlungsmechanismus (VCG-Mechanismus) der garantiert, dass keine Zahlung an Knoten erfolgt die keine Daten weiterleiten.
- Kürzeste Pfade werden anhand von c_k anstatt der Anzahl hops berechnet.
- Jeder Knoten sendet zusätzlich zu den bisherigen Informationen beim Verteilen seiner Routing-Tabelle auch seine Weiterleitungskosten (das vergrößert die BGP Routing-Tabellen lediglich um einen konstanten Faktor)
- Die lokale Berechnung wird in derselben zeitlichen Größenordnung (Komplexität) durchgeführt wie die bisherige Berechnung

Probleme hierbei:

- Ein AS ist gleichzeitig ein individuell handelnder Strategischer Agent und ein Teil des Systems zur Berechnung des Mechanismus.
- Strategiebeständige Mechanismen wurden entworfen um unwahrheitsgemäße Berichterstattung seitens der Agenten zu vermeiden.
- Wie kann man in diesem Szenario also sicherstellen, dass keiner der Agenten den Algorithmus lokal modifiziert um seinen eigenen Nutzen zu vergrößern?

3.3 Multicast Cost Sharing

Problemstellung

- Eine Population von Agenten P in einem Netzwerk N von Knoten verbunden durch eine Menge von ungerichtete Kanten L (Links).
- Der Multicast-Datenstrom entstammt von einem Quellknoten $\alpha_s \in N$
- Gegeben eine Menge von Empfängern $R \subseteq P$ fließen die Datenströme von der Quelle α_s über einen Multicast-Baum $T(R) \subseteq L$ mit der Wurzel α_s und Knoten aus R.
- Annahme: Der Multicast-Baum $T(R)$ ist der kleinste Baum $T(P)$ der benötigt wird um alle Empfänger aus R zu erreichen.
- Jedem Link $l \in L$ sind Kosten $c(l) \geq 0$ zugeordnet, die den Knoten an den Enden dieses Links bekannt sind.

- Jeder Agent i hat eine Wertschätzung v_i für das Empfangen des Multicast-Datenstroms.
- Ein Mechanismus bestimmt nun welche Agenten die Übertragung bekommen und wie viel jeder Empfänger dafür bezahlen muss (Es geht im Prinzip darum die Kosten der Übertragung gerecht auf die Empfänger zu verteilen).
- $p_i \geq 0$ sind die Kosten die Agent i bezahlen muss und σ_i gibt an, ob der Agent i die Übertragung bekommt oder nicht. $\sigma_i = 1$ wenn der Agent die Übertragung bekommt, 0 sonst.
- $v = (v_1, \ldots, v_{|P|})$ ist der Inputvektor (Nutzenvektor) der Agenten.
- Der Mechanismus ist also ein Paar von Funktionen $M(v) =< p(v), \sigma(v) >$.
- Die Menge der Empfänger ist definiert durch $R(v) = \{i | \sigma_i = 1\}$.
- Der Nutzen eines Agenten ist also definiert durch $u_i = \sigma_i v_i - p_i$ ((Wertschätzung falls der Agent die Übertragung bekommt) - Kosten).
- Die Kosten des gesamten Multicast-Netzwerkes für die Übertragung ist: $c(T(R)) = \sum_{l \in T(R)} c(l)$ (Summe der Kosten der Kanten (links) des Multicast-Baums)
- Die Summe des Nutzens aller „empfangenden" Agenten ist $v_R = \sum_{i \in R} v_i$.
- Der Nutzen des gesamten Netzwerkes ist $NW(R) = v_R - c(T(R))$ (Nutzen der Agenten - Kosten der Agenten).

Es gibt nun zwei mögliche Mechanismen zur strategiebeständigen Lösung dieses Problems, den marginal-cost und den Shapley-Value Ansatz.

- Marginal cost:
  - Strategiebeständig
  - Effizient ($NW(R)$ wird maximiert)
  - Gute Netzwerkkomplexität
  - Sei $W_i = NW(R)$ mit $i \in R$ und $W_{-i} = NW(R)$ mit $i \notin R$.
  - Kostenverteilung: $p_i = \begin{cases} 0 & wenn\ \sigma_i = 0 \\ v_i - (W_i - W_{-i}) & sonst \end{cases}$
  - Der Agent i bezahlt also den Unterschied des Nutzens des Netzwerks der durch ihn entsteht.
  - Berechenbar mit zwei (kurzen) Nachrichten pro Link und zwei (einfachen) Berechnungen pro Knoten.
  - Marginal Cost ist der einzige Mechanismus, der die folgenden zwei Eigenschaften besitzt:

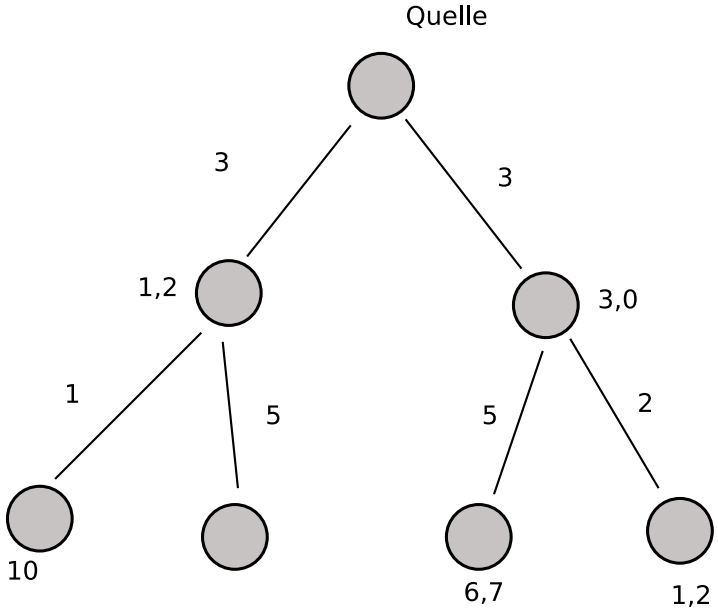

Abbildung 3: Multicast Tree

  * Keinerlei positive Transfers: $p_i(v) \geq 0$, d.h. der Mechanismus ist nicht in der Lage Agenten für das Empfangen der Multicast-Übertragung zu bezahlen.
  * Freiwillige Teilnahme: Die Agenten können durch das Setzen von $v_i = 0$ (daraus resultiert $\sigma_i = 0$) frei wählen ob sie den Datenstrom empfangen wollen oder nicht.

Der zweite Mechanismus ist der Shapley Value Ansatz. Dieser ist gruppenstrategiebeständig und budget-balanciert.

- Shapley Value
  - Gruppenstrategiebeständig
  - Budget-balanciert (Dies bedeutet, dass $\sum_{i \in R} p_i = c(T(R))$.)
  - Kostenverteilung: Die Kosten $c(l)$ eines Links werden gleich geteilt durch die Anzahl der Empfänger im Teilbaum von l. (Nicht-empfangende Agenten zahlen 0.)
  - Die Menge an Empfängern wird so groß wie möglich gewählt unter der Bedingung, dass $v_i \geq p_i$ für alle Agenten $i \in R$.
  - Schlechte Netzwerkkomplexität im Vergleich zum Marginal Cost Mechanismus

3.4 Web Caching

- Web Caches steigern die Performance von Webseitenzugriffen
  - Identifikation von „hot spots" im Netzwerk
  - Verminderung von Zugriffszeiten
- Derzeitige Web Caching Architektur:
  - Menge an kollaborativ arbeitenden Caches die eine Klientel bedienen
- Ziel: Erreichen einer globalen effizienten Lastverteilung
- Annahme: Caches verhalten sich gehorsam
- Was passiert nun, wenn ein Netzwerk von Caches sich über verschiedene Betreiber verteilt?
- Zwei Anreizprobleme:
  - Clients fragen einen einzigen Cache an
  - Caches berichten der gesamten Architektur
- Verhalten eines einzelnen Caches:
  - Maximierung des Nutzens der Benutzer
  - Cachen derjenigen Seiten welche die höchste Zugriffsrate aufweisen um den Nutzen für den Benutzer zu maximieren
  - Der Nutzen ist eine private Information der Benutzer
  - Was sind hierbei strategiebeständige Mechanismen die einen Benutzer dazu bringen wahrheitsgemäß über seinen Nutzen zu berichten?
- Kollaboration zwischen verschiedenen Caches:
  - Caches haben Kosten für das Zwischenspeichern einer bestimmten Seite
- Wenn keinerlei Zahlungen stattfinden könnten sich Caches strategisch verhalten so dass die anderen das Zwischenspeichern übernehmen[3]
- Wenn allerdings Zahlungen stattfinden, treten die Caches zueinander in Konkurrenz um diejenigen Seiten zwischenzuspeichern, die die höchste Auszahlung (d.h. den größten Nutzen für die Benutzer) bringen. So wird eine suboptimale Verteilung der gecachten Seiten, bzw. eine mehrfache Zwischenspeicherung erreicht.

[3] Team: Toll, ein anderer machts ;-)

- Ziel muss es also sein, einen Mechanismus zu entwerfen, so dass die Verteilung der Caches so optimiert wird, dass eine maximale Performance erreicht wird. Ziel ist es also den Mechanismus so zu entwerfen, dass man a) die Benutzer dazu bringt Ihren wahren Nutzen zu nennen und b) die Caches dazu bringt Ihre Ressourcen kollaborativ optimal einzusetzen.

3.5 Peer-to-Peer File Sharing

Bisherige Situation:

- Kostenloses Verteilen von Dateien (Gnutella, eMule, Kazaa, etc.)
- Eigenständige verteilte Systeme (Peers)
  - Jeder Rechner gehört einem anderen Benutzer
  - Keine zentrale administrative Instanz
- „Schenkungsgesellschaft" (Benutzer bieten Inhalte, Bandbreite und Rechenzeit kostenlos an)
- Verschiedene Arten von Nachrichten: Ping, Pong, Suche, Antworten; All diese Nachrichten werden von einem Peer an seine Nachbarn weitergeleitet

Free Rider Problem

- Niemand ist bereit die Kosten für etwas öffentlich zugängliches zu tragen wenn die Hoffnung besteht, dass jemand anders die Kosten übernehmen wird.
- In P2P Systemen gibt es viele Benutzer die lediglich herunterladen, aber selbst keinerlei Dateien zur Verfügung stellen. Dadurch ergeben sich wiederum neue Probleme, da nur einige wenige Benutzer zu einer Art „zentralen Server" werden.

Problemlösung

- Entwicklung einer Tauschwirtschaft
- Bsp: MojoNation P2P
  - Hier bekommt man „Mojos" bezahlt, wenn man Daten anbietet/verteilt.
  - Mojo kann dann auch dazu verwendet werden um sich eine bessere Position zu „erkaufen", wenn das System überlastet ist.
- Aktueller Gegenstand der Forschung: Entwicklung von *Flatratemodellen* zur legalen Verteilung digitaler Güter.
- Anderer Ansatz: Entwicklung eines Mechanismus zur verteilten Distribution digitaler Inhalte.

3.6 Distributed task allocation/CPU Marketplace

- Benutzer können CPU intensive Tasks in einen „Markt" abladen
- Zwei Möglichkeiten des „Pricings" (Zahlungsmechanismen):
  - Fixes Pricing (es werden immer dieselben Zahlungen verlangt)
  - Dynamisches Pricing (Wie bei Auktionen: die Zahlungen hängen von der derzeitigen nachfrage ab)

Einseitiges Auktionsmodell

- Gegeben: k Tasks
- n Agenten (Hosts mit Rechenkapazität; Bieter; inverse Auktion (Eine inverse Auktion hat im Gegenteil zu einer normalen Auktion mehrere Verkäufer und einen Bieter. Theoretisch betrachtet ist die inverse Auktion zu einer normalen äquivalent.))
- Typ $t_{i,j}$ des Agenten i ist die minimale Zeit des Agenten i um den Task j zu berechnen (private Information)
- x_i Menge von Tasks die Agent i zugeordnet ist.
- Nutzen des Agenten i: $-\sum_{j \in x_i} t_{i,j}$
- Ziel des Marktes: $min_i \sum_{j \in x_i} t_{i,j}$(Minimierung der Produktionsspanne (Rechendauer))
- *MinWork* Approximation der Produktionsspanne:
  - $min_i \sum_{j \in x_i} t_{i,j} \leq \sum_j min_i t_{i,j}$
  - $min_i \sum_{j \in x_i} t_{i,j} \geq (1/n) \sum_j min_i t_{i,j}$
- *MinWork* Mechanismus
  - utilitaristisch: $\sum_j min_i t_{i,j}$(inverse Auktion, die Wertschätzungen sind negativ: min anstatt max)
  - Zahlung: Ein Agent bekommt genau den Betrag gezahlt den der zweitbeste Agent für diesen Task verlangt
  - Hierbei handelt es sich wiederum um einen VCG Mechanismus. Dieser ist per Definition strategiebeständig.
  - Dieser Mechanismus ist äquivalent zu einer Vickrey-Auktion für jeden einzelnen Task
- Offene Probleme:
  - Gibt es einen *besseren* strategiebeständigen approximativen Mechanismus?

- Benutzer mit Tasks bezahlen die Agenten: Was passiert wenn mehr von Ihnen verlangt wird, also Sie zu zahlen in der Lage sind?
- →Benutzung von Mindestpreisen: Benutzer haben Mindestpreise (Wertschätzungen sind negativ) und mehr als das können Sie nicht bezahlen.
- Entwicklung eines Mechanismus unter Verwendung von Mindestpreisen

Literatur

[1] *Algorithmic mechanism design (extended abstract)*, New York, NY, USA, 1999. ACM Press.

[2] Rajdeep K. Dash, Nicholas R. Jennings, and David C. Parkes. Computational-mechanism design: A call to arms. *IEEE Intelligent Systems*, 18(6):40–47, 2003.

[3] J. Feigenbaum, C. Papadimitriou, R. Sami, and S. Shenker. A bgp-based mechanism for lowest-cost routing, 2002.

[4] J. Feigenbaum and S. Shenker. Distributed algorithmic mechanism design: Recent results and future directions. In *Proceedings of the 6th International Workshop on Discrete Algorithms and Methods for Mobile Computing and Communications*, pages 1–13. ACM Press, New York, 2002.

[5] J. Green and J.J. Laffont. *Characterization of satisfactory Mechanisms for the revelation of preferences for public goods.* Econometrica, 1977.

[6] John Hershberger and Subhash Suri. Vickrey prices and shortest paths: What is an edge worth? In *Proceedings of the 42nd IEEE Symposium on Foundations of Computer Science (FOCS'01)*, 2002.

[7] Noam Nisan and Amir Ronen. Algorithmic mechanism design. pages 129–140.