Methoden

Jan Zimmermann*, Tatiana Tatarenko, Volker Willert und Jürgen Adamy

Gradient-Tracking-basierte Lösung von Multi-Cluster-Spielen

Gradient-tracking-based solution of multi-cluster-games

https://doi.org/10.1515/auto-2021-0106
Empfangen 29. Juli 2021; angenommen 12. November 2021

Zusammenfassung: In diesem Beitrag wird die Anwendung von Gradient-Tracking-Verfahren in Multi-Cluster-Spielen untersucht. Neben einer Aufarbeitung relevanter Literatur umfasst die Arbeit einen theoretischen und simulativen Vergleich zwischen zwei bestehenden Gradient-Tracking-basierten Algorithmen. Hierbei wird insbesondere auf die Unterschiede der Kommunikationsarchitekturen eingegangen. In den durchgeführten Simulationen werden die Konvergenzzeiten der Algorithmen in Anwendung auf ein Nash-Cournot-Spiel miteinander verglichen. Es wird deutlich, dass sich eine weniger eingeschränkte Kommunikationsarchitektur zwischen den Clustern positiv auf die Konvergenzzeit auswirkt.

Schlagwörter: Verteilte Optimierung, Multi-Agenten-Systeme, Spieltheorie

Abstract: In this work, the application of gradient-tracking procedures in multi-cluster games is analyzed. Next to a reprocessing of relevant literature, the work encompasses a theoretical and simulative comparison between two existing gradient-tracking-based algorithms. Differences regarding the communication architectures are highlighted in particular. In the conducted simulations the convergence times of the algorithms are compared with each other by application of the algorithms on a Nash-cournot game. It becomes apparent that a less restricted communication architecture between the agents is beneficial regarding the convergence time.

***Korrespondenzautor: Jan Zimmermann,** Technische Universität Darmstadt, Darmstadt, Deutschland, E-Mail: jan.zimmermann@rmr.tu-darmstadt.de
Tatiana Tatarenko, Jürgen Adamy, Technische Universität Darmstadt, Darmstadt, Deutschland, E-Mails: tatiana.tatarenko@rmr.tu-darmstadt.de, adamy@rmr.tu-darmstadt.de
Volker Willert, Hochschule für angewandte Wissenschaften Würzburg-Schweinfurt, Schweinfurt, Deutschland, E-Mail: volker.willert@fhws.de

Keywords: distributed optimization, multi-agent-systems, game theory

1 Einleitung

Ein Aspekt des Smart Grid Konzepts ist die Entwicklung weg von zentralen Strukturen der Stromversorgung hin zu dezentraler Kontrolle von Zusammenschlüssen aus dezentralen Erzeugern, Speichern und Verbrauchern [1]. Solche Zusammenschlüsse werden als Microgrids bezeichnet. Um das Gesamtnetzwerk gegen Angriffe resilienter zu machen, sollten kritische Informationen des Netzwerkes nicht zentral gespeichert werden und nur dort bekannt sein, wo sie unbedingt benötigt werden. In Netzwerken, die aus Zusammenschlüssen von Microgrids bestehen, ergeben sich unterschiedliche Konkurrenz- und Kooperationssituationen. Werden die Microgrids beispielsweise von unterschiedlichen Unternehmen operiert, entsteht Konkurrenz bezüglich des zwischen den Microgrids verhandelten Strompreises und es resultieren verschiedene nicht-kooperative Energiemarkt-Szenarien [4, 6]. Innerhalb der Microgrids steht jedoch die optimale Verteilung bzw. Nutzung der vorhandenen Ressourcen im Vordergrund [2]. Werden diese Ressourcen von einem einzigen, übergeordneten Unternehmen kontrolliert, so ergibt eine kooperative Optimierung hier den größeren Vorteil. Damit entsteht am Beispiel von Microgrid-Netzwerken folgende Problematik: Auf oberer Ebene, d. h. zwischen den Microgrids, muss ein nicht-kooperatives, verkoppeltes Optimierungsproblem gelöst werden, dessen Lösung ein spieltheoretisches Gleichgewicht ist. Auf der unteren Ebene, d. h. innerhalb der Microgrids, sollte hingegen das Verteilungsproblem kooperativ optimiert werden. Für beide Probleme soll die Privatsphäre der zu optimierenden Funktionen eingehalten werden. Dieses hierarchische, verkoppelte Problem lässt sich durch agentenbasierte Multi-Cluster-Spiele modellieren. In solchen Problemstellungen einigen sich die Cluster, im Beispiel Microgrids oder Unternehmen, auf der oberen Ebene auf eine Lösung des Spiels, während simultan auf der unteren Ebene die Agenten in-

nerhalb des Clusters kooperativ das Intra-Cluster-Problem optimieren. Um trotz der verteilten Information das Gesamtproblem lösen zu können, haben die Agenten die Möglichkeit über Kommunikationskanäle bestimmte Informationen auszutauschen.

Methoden zur Lösung obiger Multi-Cluster-Probleme lassen sich in zeit-kontinuierliche und zeit-diskrete Algorithmen einteilen, wobei letztere sich mit weniger Aufwand auf Digitalrechnern umsetzen lassen. Während zu erster Gruppe bereits einige Arbeiten, z. B. [14–18, 20], veröffentlicht sind, existiert für die zweite Gruppe, mit der sich in dieser Arbeit verstärkt auseinander gesetzt wird, weniger Literatur. In der kürzlich veröffentlichten Arbeit [8] wird ein diskreter Algorithmus vorgestellt, der ohne direkte Gradienteninformation mit linearer Konvergenzrate in eine Nachbarschaft der Lösung eines unbeschränkten Multi-Cluster-Problems konvergiert. Ebenso kommt der Algorithmus aus [12] ohne Gradienteninformationen aus und verwendet ausschließlich zero-order Informationen[1] zur Aktualisierung der Zustände. Die zeit-diskreten Algorithmen in [5] und [19] verwenden Gradienteninformationen, um durch ein Gradient-Tracking-Verfahren, das ursprünglich für kooperative, verteilte Optimierung entwickelt wurde, das Optimum mit linearer Konvergenzrate zu finden. Dieses Gradient-Tracking zur Schätzung von Gradienten soll in dieser Arbeit zusammen mit den beiden genannten Algorithmen genauer untersucht werden. Die Unterschiede der Algorithmen aus [5] und [19] belaufen sich vor allem auf die Kommunikationsstruktur, für die in [5] ein ungerichteter Graph und in [19] ein gerichteter Graph mit zeilen- bzw. spalten-stochastischer Gewichtung angenommen wird. Letztere Gewichtungen stellen eine schwächere Anforderung an die Gewichtung des Graphen dar. Eine Besonderheit des Algorithmus aus [5] ist die Einführung von Cluster-Leadern und damit einer Hierarchie unter den Agenten. Dies hat eine Einschränkung der Kommunikationsstruktur zur Folge, die im Abschnitt 4 besprochen wird.

Neben einer allgemeinen Analyse von Gradient-Tracking-Verfahren umfasst der Hauptbeitrag der Arbeit einen theoretischen und simulativen Vergleich der bestehenden Algorithmen aus [5] und [19]. Insbesondere wird dabei Bezug auf die unterschiedlichen Hierarchie- und der daraus resultierenden Kommunikationsstrukturen genommen und die theoretischen Erkenntnisse durch die Simulation eines erweiterten Nash-Cournot-Spiels untermauert. Bei dem Artikel [19] handelt es sich um eine frühere Arbeit der Autoren dieses Beitrags.

1 Zero-order Informationen bedeutet, dass die funktionale Form der Kostenfunktion dem Optimierer unbekannt ist und der Wert der Kostenfunktion nur an einzelnen Stützstellen abgefragt werden kann.

Die Arbeit gliedert sich wie folgt: Zunächst werden in Abschnitt 2 die Notation und grundlegende Begriffe bezüglich Kommunikationsgraphen erläutert. Anschließend wird in Abschnitt 3 das Problem der Multi-Cluster-Spiele formuliert. In Abschnitt 4 wird auf die Lösung von Multi-Cluster-Spielen durch Gradient-Tracking-Verfahren eingegangen und die Algorithmen vorgestellt, bevor in Abschnitt 5 die Simulationsergebnisse dargelegt werden. In Abschnitt 6 wird schließlich das Fazit gezogen.

2 Notation und Kommunikationsgraphen

In dieser Arbeit werden Matrizen $\boldsymbol{M}$ durch Fettdruck von Vektoren v und Skalaren s unterschieden. Eine Ausnahme bildet die Notation $\mathbf{1}$, die einen Vektor mit Einträgen $\mathbf{1}_i = 1$ beschreibt. Verkürzt wird für den Gradienten ausgewertet an der Stelle x_r $\nabla_x f(x,y)|_{x=x_r}$ die Notation $\nabla_x f(x_r, y)$ verwendet. Die Vorschrift $\boldsymbol{x} = (x_i)_{i=1}^n$ stapelt die n liegenden Vektoren $x_i \in \mathbb{R}^{1\times q}$ zu einer Matrix $\boldsymbol{x} \in \mathbb{R}^{n\times q}$. Ein Kommunikationsgraph eines Agentensystems $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$ besteht aus der Knotenmenge $\mathcal{V}$ und der Kantenmenge $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$. Die Knoten repräsentieren die Agenten und die Kanten die Kommunikationskanäle des Systems. Kann Agent i an Agent j Informationen schicken, so gilt $(i,j) \in \mathcal{E}$. Handelt es sich um ungerichtete Verbindungen, gilt: Wenn $(i,j) \in \mathcal{E}$, dann auch $(j,i) \in \mathcal{E}$. Diese Bidirektionalität gilt für gerichtete Graphen im Allgemeinen nicht. Agent j zählt zur Nachbarschaft $\mathcal{N}_i$ von Agent i, wenn i von j Informationen empfangen kann. Zusätzlich gilt $i \in \mathcal{N}_i$. Ein ungerichteter Graph wird als verbunden bezeichnet, wenn ein ungerichteter Pfad von einem beliebigen Agenten zu jedem anderen Agenten im Netzwerk existiert. Bei gerichteten Graphen spricht man von starker Verbundenheit, wenn es solch einen gerichteten Pfad zwischen den Agenten gibt. Existiert eine Kante (i,j), wird ihr Gewicht mit $a_{ji} > 0$ beschrieben, existiert sie nicht, so ist ihr Gewicht gleich null, $a_{ji} = 0$. Zusammengefasst werden diese Gewichte in der Adjazenzmatrix $\boldsymbol{A} \in \mathbb{R}^{|\mathcal{V}|\times|\mathcal{V}|}$ des Graphen, wobei für die Eigengewichte der Agenten $a_{ii} > 0,\ \forall i$ gilt. Die Adjazenzmatrix $\boldsymbol{A}$ ist zeilen-stochastisch, wenn $\boldsymbol{A}\mathbf{1} = \mathbf{1}$ und spalten-stochastisch, wenn $\mathbf{1}^T\boldsymbol{A} = \mathbf{1}^T$. Gilt sowohl $\boldsymbol{A}\mathbf{1} = \mathbf{1}$ als auch $\mathbf{1}^T\boldsymbol{A} = \mathbf{1}^T$, wird die Matrix als doppelt-stochastisch bezeichnet.

3 Problemformulierung

Es sei ein Agentensystem gegeben, das insgesamt aus n Agenten besteht. Diese Agenten sind innerhalb des Sys-

tems verschiedenen Agentengruppen zugeordnet, die als Cluster bezeichnet werden. Insgesamt existieren H Agentencluster im System. Die Agentenmengen $\mathcal{A}_h$ aller Cluster $h = 1, ..., H$ sind zueinander disjunkt, $\mathcal{A}_i \cap \mathcal{A}_j = \emptyset$ für alle $i, j = 1, ..., H$, $i \neq j$, sodass jeder Agent nur einem einzigen Cluster zugeordnet ist. Darüber hinaus ist jeder Agent Teil eines Clusters, $i \in \mathcal{A}_1 \cup ... \cup \mathcal{A}_H$, und alle Cluster umfassen mindestens einen Agenten, $\mathcal{A}_h \neq \emptyset$. Die Anzahl der Agenten innerhalb eines Clusters wird durch $|\mathcal{A}_h| = n_h$ beschrieben.

Jeder Agent i des Clusters h besitzt eine private Kostenfunktion $f_i^h(x^h, x^{-h})$, die von der von allen Agenten im Cluster h geteilten Clusteraktion $x^h \in \mathbb{R}^{q_h}$ abhängt, sowie von den Aktionen aller anderen Cluster $x^{-h} \in \mathbb{R}^{q-q_h}$ ausschließlich Cluster h, wobei q die Dimension des zusammengesetzten Aktionsvektors $x = [x^h, x^{-h}]$ ist. Die Agenten innerhalb eines Clusters kooperieren miteinander, um die Summe aus ihren Kostenfunktionen $F^h(x) = \sum_{i=1}^{n_h} f_i^h(x^h, x^{-h})$ zu minimieren, indem die Clusteraktion x^h entsprechend angepasst wird. Die Cluster selbst kooperieren allerdings nicht miteinander und verfolgen egoistisch das Ziel der eigenen Agentengemeinschaft. Da die Kostenfunktionen der Agenten in Cluster h auch von den Aktionen der anderen Cluster beeinflusst werden, befinden sich die Cluster in einem nicht-kooperativen Spiel $\Gamma(H, \mathbb{R}^q, \{F^h\})$ zueinander, dessen Lösung ein Nash-Gleichgewicht ist. Damit lässt sich das Optimierungsproblem eines unbeschränkten Multi-Cluster-Spiels wie folgt formulieren:

$$\min_{x^h} F^h(x^h, x^{-h}) = \min_{x^h} \sum_{i=1}^{n_h} f_i^h(x^h, x^{-h}), \ \forall h. \tag{1}$$

Entscheidend ist hier, dass jeder Cluster h nur seine eigene Aktion x^h anpassen kann und nicht die Aktionen der restlichen Cluster. Bevor aus den beschriebenen Zielen des Multi-Cluster-Spiels Optimalitätsbedingungen für potentielle Lösungen generiert werden, werden folgende Annahmen bezüglich der Kostenfunktionen und der Spiel-Abbildung getroffen:

Annahme 1. *Die lokalen Kostenfunktionen $f_i^h(x^h, x^{-h})$ sind konvex in x^h für alle $i = 1, ..., n_h$ und $h = 1, ..., H$, kontinuierlich differenzierbar und ihr Gradient $\nabla_{x^h} f_i^h(x^h, x^{-h})$ ist Lipschitz-kontinuierlich auf $\mathbb{R}^{q_h}$.*

Annahme 2. *Die Spiel-Abbildung $\boldsymbol{M}(x) : \mathbb{R}^q \to \mathbb{R}^q$*

$$\boldsymbol{M}(x) = [g^1(x)^T, ..., g^H(x)^T]^T \tag{2}$$

mit $g^h(x^h, x^{-h}) \triangleq \sum_{i=1}^{n_h} \nabla_{x^h} f_i^h(x^h, x^{-h}) \in \mathbb{R}^{q_h}$ ist gleichmäßig monoton auf $\mathbb{R}^q$ mit einer Konstanten $\mu > 0$, d. h., es gilt $(\boldsymbol{M}(x) - \boldsymbol{M}(y))^T (x - y) \geq \mu ||x - y||^2$.

Anmerkung 1. *Gleichmäßige Monotonie des Spielmappings $\boldsymbol{M}(x)$ impliziert gleichmäßige Konvexität der Cluster-Kostenfunktionen $F^h(x^h, x^{-h})$ in x^h für feste x^{-h}.*

3.1 Optimalitätsbedingungen

Angelehnt an [19] werden zunächst zwei Optimalitätsbedingungen formuliert. Der Vektor $(x^{\text{opt}})^h$ ist eine optimale Lösung des kooperativen Optimierungsproblems innerhalb des Clusters h, bei konstanten Aktionen x^{-h} der restlichen Cluster, wenn Annahme 1 gilt und die Bedingung

$$\nabla_{x^h} F^h((x^{\text{opt}})^h, x^{-h}) = \sum_{i=1}^{n_h} \nabla_{x^h} f_i^h((x^{\text{opt}})^h, x^{-h}) = 0 \tag{3}$$

erfüllt ist. Daneben ist das Nash-Gleichgewicht des nicht-kooperativen Spiels $\Gamma(H, \mathbb{R}^q, \{F^h\})$ erreicht, wenn für den Vektor $x^{\text{NE}} \in \mathbb{R}^q$ die Bedingung

$$F^h\left((x^{\text{NE}})^h, (x^{\text{NE}})^{-h}\right) \leq F^h\left(x^h, (x^{\text{NE}})^{-h}\right), \forall h \tag{4}$$

gilt.

Tatsächlich ist es möglich, beide Optimalitätsbedingungen zu einer zu kombinieren. Ist Annahme 2 erfüllt, existiert nur ein einziges, eindeutiges Nash-Gleichgewicht und in diesem gilt für die Spiel-Abbildung $\boldsymbol{M}(x^{\text{NE}}) = \boldsymbol{0}$ [11]. Das bedeutet, dass im Nash-Gleichgewicht x^{NE} für die Gradienten $\sum_{i=1}^{n_h} \nabla_{x^h} f_i^h((x^{\text{NE}})^h, (x^{\text{NE}})^{-h}) = 0$ für $h = 1, ..., H$ gilt, was die Bedingung (3) beinhaltet. Damit ist das Multi-Cluster-Spiel gelöst, wenn

$$\nabla_{x^h} F^h(x^*) = \sum_{i=1}^{n_h} \nabla_{x^h} f_i^h((x^*)^h, (x^*)^{-h}) = 0 \tag{5}$$

für alle $h = 1, ..., H$ erfüllt ist, mit $x^* = [(x^{\text{opt}})^h, (x^{\text{opt}})^{-h}] = x^{\text{NE}}$. Eine ähnliche Formulierung findet sich auch in [5].

Um in der Lage zu sein, den Gradienten seiner Kostenfunktion auswerten zu können, schätzt jeder Agent i den globalen Aktionsvektor x in der Variablen x_i. Die Schätzungen aller Agenten müssen sich über die Zeit hinweg aneinander annähern und schließlich einen Konsensus bilden, d. h.

$$x_i = x_j = x_c, \text{ für alle } i, j = 1, ..., n, \tag{6}$$

der bei Konvergenz des Verfahrens gleich der optimalen Lösung x^* ist. Neben Informationen zu Schätzung der Aktionen werden keinerlei weitere Informationen bezüglich des Optimierungsproblems zwischen den konkurrierenden Clustern ausgetauscht.

3.2 Sonderfälle von Multi-Cluster-Spielen

Dieser Abschnitt widmet sich zwei Sonderfällen, die an den Rändern des Definitionsbereiches von Multi-Cluster-Spielen liegen und sich zu bekannten Problemstellungen reduzieren. Zunächst wird der Fall betrachtet, dass nur ein einziger Cluster im Agentensystem existiert, d. h. $H = 1$, der alle n Agenten beinhaltet. Da für diesen Cluster kein Gegenspieler existiert, fällt der nicht-kooperative Anteil des Multi-Cluster-Spiels weg und das Problem reduziert sich auf das weithin bekannte rein kooperative, verteilte Optimierungsproblem der Form

$$\min_x F^1(x) = \min_x \sum_{i=1}^{n} f_i^1(x). \tag{7}$$

Ein weiterer Sonderfall entsteht, wenn es mehr als einen Cluster im Netzwerk gibt, $H > 1$, allerdings jeder dieser Cluster nur einen einzigen Agenten umfasst, d. h. $n_h = 1, \forall h$. In diesem Fall entfällt der kooperative Teil des Multi-Cluster-Spiels, da sich die Cluster-Kostenfunktion auf eine einzige, bekannte Kostenfunktion reduziert, für die der Gradient direkt berechnet werden kann. Entsprechend verbleibt nur das nicht-kooperative Spiel zwischen den Clustern, sodass die allgemeine Form eines Nash-Equilibrium-Problems

$$\min_{x^h} f_1^h(x^h, x^{-h}), \forall h$$

resultiert. Damit lässt sich festhalten, dass Multi-Cluster-Spiele sich sowohl als Kombination von nicht-kooperativen Spielen und verteilter, kooperativer Optimierung verstehen lassen sowie als Verallgemeinerung dieser Sonderfälle. Daraus folgt, dass Algorithmen für Multi-Cluster-Spiele auch Nash-Gleichgewichtsprobleme und kooperative, verteile Optimierungsprobleme lösen können, vorausgesetzt, diese Probleme erfüllen die Annahmen 1 bzw. 2.

4 Lösung von Multi-Cluster-Spielen

Für eine effiziente Beschreibung der Algorithmen zur Lösung von Multi-Cluster-Spielen werden zunächst, an [19] angelehnt, verschiedene Variablen und Vektoren definiert.

$k = 0, 1, \ldots$ beschreibe einen diskreten Zeitpunkt. Die Schätzung des Agenten i zum Zeitpunkt k bezüglich des Aktionsvektors x wird deklariert als

$$x_i(k) = \left[x_i^1(k), \ldots, x_i^h(k), \ldots, x_i^H(k)\right] \in \mathbb{R}^{1\times q},$$

wobei $x_i^h(k)$ der Schätzung der Aktionsvariablen des Clusters h entspricht. Neben der Schätzvariablen x_i benötigen die Agenten eine zusätzliche Variable $y_i^h(k) \in \mathbb{R}^{1\times q_h}$, wenn $i \in \mathcal{A}_h$, wobei $\dim(y_i^h) = \dim(x^h)$. Diese Variable wird als Gradient-Tracking-Variable des Agenten i bezeichnet, die verwendet wird, um den Aktionsvektor x_i^h des Agenten i in Cluster h zu aktualisieren. Um diese Schätzung $y_i^h(k) \in \mathbb{R}^{1\times q_h}$ in Kombination mit der Schätzvariablen $x_i(k) \in \mathbb{R}^{1\times q}$ verwenden zu können, wird der Vektor mit Nullen aufgefüllt, sodass

$$\hat{y}_i^h(k) = [0_{1\times q_{<h}}, y_i^h(k), 0_{1\times q_{>h}}] \in \mathbb{R}^{1\times q}, \tag{8}$$

mit $q_{<h} = \sum_{l=1}^{h-1} q_l$ und $q_{>h} = \sum_{l=h+1}^{H} q_l$. Durch diese Vorschrift beinhaltet die Variable $\hat{y}_i^h(k)$ nur an denjenigen Stellen Gradient-Tracking-Informationen, an denen der Agent i in der Variablen $x_i(k)$ die Aktionen des eigenen Clusters h schätzt.

4.1 Gradient-Tracking: Von verteilter Optimierung zu Multi-Cluster-Spielen

Gradient-Tracking-Verfahren wurden ursprünglich für verteilte, kooperative Optimierung, d. h. für den Spezialfall $H = 1$ in Gleichung (7) von Multi-Cluster-Spielen entwickelt, siehe z. B. [7, 10, 13]. In diesem Abschnitt wird die grundlegende Funktionsweise der Gradientenschätzung durch das Gradient-Tracking vermittelt und erklärt, wie sich das Verfahren für Multi-Cluster-Spiele erweitern lässt.

Es sei dafür zunächst ein Agentensystem mit n Agenten gegeben, die sich in einem einzigen Cluster $h = 1$ befinden. Untereinander sind die Agenten durch einen ungerichteten, verbundenen Graphen $\mathcal{G}$ miteinander vernetzt, die gewichtete Adjazenzmatrix $\boldsymbol{A} \in \mathbb{R}^{n\times n}$ sei doppeltstochastisch. Jeder Agent i schätzt in der Variablen $y_i(k)$ den Gradienten der Funktion $F^1(x_i(k))$, um mit dieser Schätzung einen Schritt bezüglich der Aktion $x_i(k)$ in Richtung des Optimums x^{opt} durchzuführen. Jeder Agent aktualisiert hierfür seine eigene Schätzung $y_i(k)$ unter Verwendung des Gradienten seiner lokalen Kostenfunktion f_i^1 [7, 10, 13]

$$\begin{aligned} y_i(k+1) = &\sum_{j=1}^{n_h} \boldsymbol{A}_{ij} y_j(k) \\ &+ \nabla_x f_i^1(x_i(k+1)) - \nabla_x f_i^1(x_i(k)). \end{aligned} \tag{9}$$

Initialisiert wird die Variable durch eine zufällige Startschätzung von $x_i(0)$ und

$$y_i(0) = \nabla_x f_i^1(x_i(0)), \text{ für } i = 1, \ldots, n. \tag{10}$$

Die nachfolgenden Überlegungen stützen sich lose auf [7]. Durch Initialisierung mit (10) ergibt sich für die Schätzungen $\boldsymbol{y}(k) = (y_i(k))_{i=1}^n$ des ersten Zeitschritts

$$\mathbf{1}^T\boldsymbol{y}(1) = \mathbf{1}^T\boldsymbol{A}\boldsymbol{y}(0) + \sum_{i=1}^{n} \nabla_x f_i^1(x_i(1)) - \sum_{i=1}^{n} \nabla_x f_i^1(x_i(0)).$$

Aufgrund der Annahme, dass $\boldsymbol{A}$ doppelt-stochastisch ist, d. h. $\mathbf{1}^T\boldsymbol{A} = \mathbf{1}^T$, und unter Verwendung der Initialisierung (10) ergibt sich

$$\mathbf{1}^T\boldsymbol{y}(1) = \sum_{i=1}^{n} \nabla_x f_i^1(x_i(1)).$$

Das bedeutet, dass die Summe über die Schätzvariablen im ersten Zeitschritt gleich der Summe der Gradienten der Kostenfunktionen der Agenten an ihren unterschiedlichen Zustandsschätzungen $x_i(1)$ ist. Tatsächlich lässt sich durch Induktion zeigen, dass dies in jedem Zeitschritt der Fall ist, d. h., es gilt

$$\mathbf{1}^T\boldsymbol{y}(k) = \sum_{i=1}^{n} \nabla_x f_i^1(x_i(k)). \tag{11}$$

Nun wird der Fall angenommen, dass die Schätzung der Aktion x_i ab dem Zeitpunkt k' konstant bleibt, d. h., es gilt $x_i = x_j = x_c$ für alle i, j. Damit gilt für den Endwert der Matrix $\boldsymbol{y}(k)$, resultierend aus der doppelten Stochastik der Matrix $\boldsymbol{A}$,

$$\boldsymbol{y}(\infty) = \frac{1}{n}\mathbf{1}\mathbf{1}^T\boldsymbol{y}(k') = \frac{1}{n}\mathbf{1}\sum_{i=1}^{n} \nabla_x f_i^1(x_c), \tag{12}$$

wodurch sich der Konsensuswert $y_i = y_j = y_c = \frac{1}{n}\sum_{i=1}^{n} \nabla_x f_i^1(x_c)$ für alle i, j ergibt. Damit ist gezeigt, dass die Gradienten-Tracking-Variable y_i jedes Agenten gegen den Mittelwert der Gradienten der Kostenfunktionen der einzelnen Agenten konvergiert, der dem Gesamtgradienten der Kostenfunktion des Problems aus Gleichung (7) entspricht, skaliert mit dem Faktor $1/n$.

Die vorausgegangenen Überlegungen sollen eine Intuition bezüglich der Funktionsweise des Gradient-Trackings vermitteln. Selbstverständlich müssen die Aktualisierungsgleichungen von Aktionsvariablen, die hier ausgelassen sind, und Gradienten-Tracking-Variablen gemeinsam in Betracht gezogen werden, um die Konvergenz des Algorithmus nachzuweisen. Dennoch lassen sich aus diesen Überlegungen Erkenntnisse für das Generieren eines Algorithmus für Multi-Cluster-Spiele gewinnen, worauf im Folgenden eingegangen wird.

Gegeben sei nun ein Multi-Cluster-Spiel, wie es in Abschnitt 3 definiert ist, und die Annahmen 1 und 2 seien erfüllt. Das Spiel gilt als gelöst, wenn neben der Konsensusbedingung (6) das Gradientenkriterium (5) erfüllt ist. Dieses Kriterium kann so interpretiert werden, dass, wenn sich alle Cluster gleichzeitig in ihrem lokalen Optimum befinden, das Nash-Gleichgewicht erreicht und das Spiel gelöst ist. Um diese lokalen Optima in jedem Cluster h zu finden, müsste im nicht-verteilten Fall mit voller Information die Summe der Gradienten $\sum_{i=1}^{n_h} \nabla_{x^h} f_i^h(x^h, x^{-h})$ in jedem Cluster bekannt sein, um einen Gradientenabstieg in die korrekte Richtung durchzuführen. Da im verteilten Fall die Kostenfunktionen privat bleiben sollen, muss die Summe der Gradienten geschätzt werden, wofür, wie bereits im vorherigen Abschnitt motiviert, ein Gradient-Tracking-Verfahren verwendet werden kann. Wie in Gleichung (12) festgestellt, ist es möglich, mit diesem Ansatz den mittleren Gradienten in einem Agentennetzwerk zu schätzen. Im Fall von Multi-Cluster-Spielen stellen allerdings die Cluster den Agenten von konkurrierenden Clustern keine Gradienteninformationen zu Verfügung, d. h. jeder Agent schätzt ausschließlich den Gradienten des eigenen Clusters, womit eine Anpassung notwendig wird.

In der Aktualisierungsgleichung der Gradientenschätzung in (9) vernetzt die Matrix $\boldsymbol{A}$ alle Agenten des Systems miteinander. Wenn nun diese Matrix zerlegt wird, indem alle Kommunikationsverbindungen zwischen Agenten, die sich nicht in einem gemeinsamen Cluster befinden, gekappt werden, entsteht für jeden Cluster h ein lokaler Kommunikationsgraph $\boldsymbol{A}^h$, der ausschließlich diejenigen Agenten verbindet, die sich gemeinsam in diesem Cluster h befinden. Beispiele solcher lokaler Graphen sind in Abbildung 3 dargestellt. Wird zusätzlich sichergestellt, dass diese lokalen Kommunikationsgraphen bestimmte Annahmen bezüglich Verbundenheit und Gewichtung erfüllen (siehe Abschnitte 4.2 und 4.3), ist es möglich, clusterbezogene Gradientenschätzungen y_i^h für $i \in \mathcal{A}^h$ durchzuführen, für die bei entsprechender Initialisierung der Zusammenhang

$$\mathbf{1}^T y^h(k) = \sum_{i=1}^{n_h} \nabla_{x^h} f_i^h(x_i(k)) \tag{13}$$

gilt. Diese Schätzungen können anschließend verwendet werden, um die Cluster-Aktionen zu aktualisieren.

Somit kann durch geschickte Einschränkung der Kommunikation, bei zusätzlicher Anpassung der Aktualisierungsgleichung der Aktionsvariablen, erreicht werden, dass nicht mehr ein Cluster in Richtung seines Optimums konvergiert, sondern alle Cluster ihre eigenen, nicht-kooperativen Ziele erreichen, die zusammengenommen die Lösung des Spiels darstellen. Der tatsächliche Konvergenzbeweis muss zusätzlich die Verzahnung

von Aktualisierungsgleichungen der Aktion, der Gradientenschätzung sowie die Konsensusdynamik berücksichtigen. Im Folgenden werden zwei Algorithmen vorgestellt, für die den jeweiligen Autoren dieser Beweis gelungen ist.

4.2 Algorithmus 1: Hierarchische Cluster

In [5] ist ein Algorithmus veröffentlicht, der auf dem beschriebenen Gradient-Tracking-Verfahren beruht. Im Gegensatz zum zweiten Verfahren, welches in Abschnitt 4.3 vorgestellt wird, besteht in diesem eine Hierarchie unter den Agenten eines Clusters: In jedem Cluster $h = 1, ..., H$ nimmt ein Agent die Rolle des Cluster-Leaders ein, der im Gegensatz zu den restlichen Agenten im Cluster die Möglichkeit hat, über die Clustergrenzen hinaus mit den Leadern anderer Cluster zu kommunizieren. Die Cluster-Leader sind über den Kommunikationsgraphen $\mathcal{G}^{\mathrm{L}}$ vernetzt, der im Algorithmus durch die gewichtete Adjazenzmatrix $\boldsymbol{A}^{\mathrm{L}}$ repräsentiert wird. Unabhängig von der Clusterzugehörigkeit werden alle Leader der Agentenmenge $\mathcal{A}_{\mathrm{L}}$ zugeordnet. Die restlichen Agenten in einem generischen Cluster h sind über einen weiteren, lokalen Kommunikationsgraphen $\mathcal{G}^h$, repräsentiert durch die gewichtete Adjazenzmatrix $\boldsymbol{A}^h$, mit ihrem jeweiligen Leader und untereinander vernetzt. Beispielhaft ist der Aufbau der Kommunikationsstruktur in Abbildung 1 dargestellt. Bezüglich der Graphen wird in [5] folgende Annahme getroffen.

Annahme 3. *Die lokalen Graphen $\mathcal{G}^h$, $h = 1, ..., H$ sind ungerichtet und verbunden. Die zugehörigen Adjazenzmatrizen $\boldsymbol{A}^h \in \mathbb{R}^{n_h \times n_h}$, $h = 1, ..., H$, sind doppelt-stochastisch.*

Ebenso ist der Leadergraph $\mathcal{G}^{\mathrm{L}}$ ungerichtet und verbunden. Dessen gewichtete Adjazenzmatrix $\boldsymbol{A}^{\mathrm{L}} \in \mathbb{R}^{H \times H}$ ist doppelt-stochastisch.

Zur Lösung des Multi-Cluster-Spiels initialisieren die Agenten ihre Schätzungen der Aktionsvariablen zufällig und setzen dann $y_i^h(0) = \nabla_{x^h} f_i^h(x(0))$ als den Startwert ihrer Gradientenschätzung. In jeder Iteration aktualisiert jeder Agent i in Cluster h seine Schätzung unter Verwendung der unter Verwendung der Aktualisierungsvorschriften aus Algorithmus 1.

Algorithmus 1.

$$x_i(k+1) = \begin{cases} \frac{1}{2}\sum_{l=1}^{n_{\mathrm{L}}} \boldsymbol{A}_{il}^{\mathrm{L}} x_l(k) + \frac{1}{2}\sum_{j=1}^{n_h} \boldsymbol{A}_{ij}^h x_j(k) - \alpha \hat{y}_i^h(k), \\ \qquad\qquad\qquad \text{wenn } i \in \mathcal{A}_{\mathrm{L}}, \\ \sum_{j=1}^{n_h} \boldsymbol{A}_{ij}^h x_j(k) - \alpha \hat{y}_i^h(k), \quad \text{wenn } i \notin \mathcal{A}_{\mathrm{L}}, \end{cases} \tag{14a}$$

$$\begin{aligned} y_i^h(k+1) = &\sum_{j=1}^{n_h} \boldsymbol{A}_{ij}^h y_j^h(k) \\ &+ \nabla_{x^h} f_i^h(x_i^h(k+1), x_i^{-h}(k+1)) \\ &- \nabla_{x^h} f_i^h(x_i^h(k), x_i^{-h}(k)). \end{aligned} \tag{14b}$$

Wie zu erkennen ist, wird in der Aktualisierungsvorschrift der Aktionen x_i zwischen Cluster-Leader-Agenten und Agenten unterschieden. Ein Agent $i \notin \mathcal{A}_{\mathrm{L}}$ aktualisiert seine Aktionsschätzung durch Bildung einer gewichteten Summe aus seiner eigenen Schätzung $x_i(k)$ und aus den Schätzungen der restlichen Cluster-Nachbarn, d. h. $x_j(k)$, $j \in \mathcal{N}_i/\{i\}$. Zusätzlich wird ein Schritt in Richtung der Gradientenschätzung $\alpha \hat{y}_i^h(k)$ unter Verwendung einer konstanten Schrittweite α durchgeführt. Ein Cluster-Leader bildet zusätzlich eine gewichtete Summe aus seiner Schätzung $x_i(k)$ und der Schätzung der anderen Cluster-Leader $x_l(k)$, $l \in \mathcal{A}_{\mathrm{L}}/\{i\}$, und führt diese mit der gewichteten Summe der lokalen Schätzungen mit jeweils einem Faktor von 1/2 zusammen. Die Variable $\hat{y}_i^h(k)$ ist in Gleichung (8) definiert und stellt sicher, dass nur die Schätzung der Variablen des eigenen Clusters x_i^h von dem Gradientenschritt verändert wird, während für x_i^{-h} keine Änderung erfolgt.

Die Aktualisierungsgleichung der Gradientenschätzung ergibt sich, wie im vorherigen Abschnitt motiviert, unter Verwendung von lokalen Graphen $\mathcal{G}^h$ bzw. Adjazenzmatrizen $\boldsymbol{A}^h$ und den jeweiligen Gradienten der lokalen Kostenfunktionen, ausgewertet an $x_i(k+1)$ und $x_i(k)$. Wie in [5] gezeigt, konvergiert obiger Algorithmus unter Verwendung einer konstanten Schrittweite α linear gegen das Optimum x^*.

Durch die Einführung einer Hierarchie unter den Agenten innerhalb eines Clusters ergibt sich eine Einschränkung der Inter-Cluster-Kommunikation. Jeder Cluster kann nach [5] nur einen Cluster-Leader besitzen und ausschließlich dieser Cluster-Leader kann mit anderen Clustern kommunizieren. In Abbildung 1 ist der Leader-Graph zusammen mit den lokalen Graphen dargestellt.

4.3 Algorithmus 2: Leader-freie Cluster

Der in [19] veröffentlichte Algorithmus fußt auf dem sogenannten Push-Pull-Gradient-Tracking-Verfahren von [9], welches ursprünglich für kooperative, verteilte Optimierungsprobleme veröffentlicht wurde. Unter Anwendung der Überlegungen aus Abschnitt 4.1 wird dieser Algorithmus in [19] für die Anwendung auf Multi-Cluster-Spiele erweitert und lineare Konvergenz nachgewiesen. Während

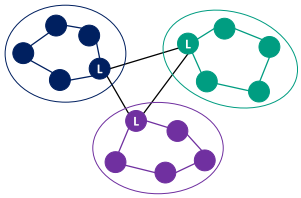

Abb. 1: Kommunikationsgraphen $\mathcal{G}^{\mathrm{L}}$ zwischen Leadern L und $\mathcal{G}^h$ in Aktualisierungsgleichung (14a) in Algorithmus 1.

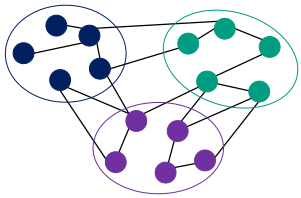

Abb. 2: Globaler Kommunikationsgraph $\mathcal{G}$ in Aktualisierungsgleichung (15a) in Algorithmus 2.

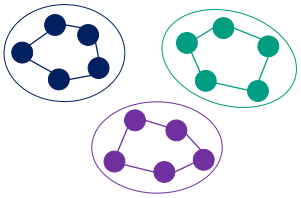

Abb. 3: Lokale Kommunikationsgraphen $\mathcal{G}^h$ in Aktualisierungsgleichung (14b) bzw. (15b) in Algorithmus 1 bzw. 2.

der im vorherigen Abschnitt beschriebene Algorithmus eine Hierarchie unter den Agenten innerhalb eines Clusters benötigt, was die beschriebene Einschränkung der globalen Kommunikation zur Folge hat, setzt der Algorithmus aus [19] nicht die Existenz von Cluster-Leadern voraus und erlaubt damit eine allgemeinere Kommunikation zwischen den Clustern: Der globale Kommunikationsgraph $\mathcal{G}$ verbindet die Agenten unabhängig von der Clusterzugehörigkeit, wie in Abbildung 2 dargestellt. Wie später gezeigt wird, kann die Leader-Kommunikationsstruktur als Spezialfall der allgemeineren Kommunikation ohne Leader verstanden werden. Zusätzlich wird für den Algorithmus aus [19] keine ungerichtete Kommunikation gefordert, was ebenso eine Verallgemeinerung darstellt. Folgende Annahme wird bezüglich der lokalen und globalen Kommunikation in [19] getroffen:

Annahme 4. *Die lokalen Graphen $\mathcal{G}^h \in \mathbb{R}^{n_h \times n_h}$, $h = 1, \dots, H$, sind gerichtet und stark verbunden. Die zugehörigen gewichteten Adjazenzmatrizen $\boldsymbol{C}^h$ sind spalten-stochastisch.*

Der globale Graph $\mathcal{G}$ ist gerichtet und stark verbunden. Die zugehörige, gewichtete Adjazenzmatrix $\boldsymbol{R} \in \mathbb{R}^{n \times n}$ ist zeilen-stochastisch.

Die Aktualisierungsvorschriften für die Agenten sind in [19] folgendermaßen festgelegt:

Algorithmus 2.

$$x_i(k+1) = \sum_{j=1}^{n} \boldsymbol{R}_{ij}(x_j(k) - \alpha \hat{y}_j^h(k)), \tag{15a}$$

$$\begin{aligned} y_i^h(k+1) = &\sum_{j=1}^{n_h} \boldsymbol{C}_{ij}^h y_j^h(k) \\ &+ \nabla_{x^h} f_i^h(x_i^h(k+1), x_i^{-h}(k+1)) \\ &- \nabla_{x^h} f_i^h(x_i^h(k), x_i^{-h}(k)), \end{aligned} \tag{15b}$$

wobei auch hier α eine positive, konstante Schrittweite ist. Die Struktur dieser Gleichungen ähnelt denen des Push-Pull-Gradient-Tracking Algorithmus aus [9], allerdings wird, wie in Abschnitt 4.1 motiviert, die Aktualisierung der Gradientenschätzung in Gleichung (15b) auf Intra-Cluster-Kommunikation beschränkt und zusätzlich die Variablen $\hat{y}_j^h(k)$, definiert in Gleichung (8), eingeführt, sodass durch den Gradientenschritt ausschließlich die Variable des eigenen Clusters aktualisiert wird.

Neben der Hierarchie ist ein Unterschied zu Algorithmus 1, dass hier in der Aktualisierung der Aktion (15a) zuerst die Gradientenaktualisierung durchgeführt wird und danach die Differenz kommuniziert wird, während in (14a) zuerst kommuniziert und anschließend die Gradienteninformation verwendet wird.

4.4 Diskussion der Kommunikationsstrukturen

In den Abbildungen 1 bis 3 sind beispielhaft die Kommunikationsgraphen der beiden beschriebenen Algorithmen dargestellt. Um die Strukturen besser vergleichen zu können, wird hier davon ausgegangen, dass auch die Kommunikation für den Algorithmus aus [19] ungerichtet ist, obwohl in der Quelle Konvergenz auch für gerichtete Graphen nachgewiesen wurde. In Abbildung 3 sind die lokalen Graphen $\mathcal{G}^h$ dargestellt, die ausschließlich Agenten innerhalb der Cluster verbinden. Hierfür überschneiden sich die Definitionen beider Algorithmen. Unterschiede lassen sich allerdings, wie bereits angesprochen, bei der globalen Kommunikation finden. Aus Abbildung 1 wird ersichtlich, dass der Kommunikationsgraph, der für die Aktualisierung der Aktionsschätzungen in Algorithmus 1 verwendet wird, sich aus dem Leadergraph $\mathcal{G}^{\mathrm{L}}$ und den lokalen Kom-

munikationsgraphen $\mathcal{G}^h$ zusammensetzt. Das heißt, dass die lokalen Kommunikationsgraphen sowohl für die Aktualisierung der Gradientenschätzung als auch für die Aktualisierung der Aktionsvariablen verwendet werden. Wie in Abbildung 2 dargestellt, müssen für Algorithmus 2 die lokalen Kommunikationsgraphen nicht zwangsweise ein Bestandteil des globalen Kommunikationsgraphen sein. Einzige Anforderung an den globalen Graphen ist hier, dass dieser verbunden bzw., im Fall der gerichteten Kommunikation, stark verbunden ist. Dies ist eine weitere Verallgemeinerung der Graphenstruktur im Vergleich zu [5].

5 Simulation

5.1 Nash-Cournot Multi-Cluster-Spiel

Nash-Cournot-Spiele sind weithin bekannte Beispiele für ökonomische, spieltheoretische Probleme. Die Formulierung der Multi-Cluster-Problemstellung orientiert sich an [19]. In einem Nash-Cournot Spiel konkurriert eine bestimmte Anzahl von Unternehmen, die alle das gleiche Produkt herstellen, bezüglich des Verkaufs dieses Produkts an einen Verbraucher. Jedes Unternehmen h verfügt über insgesamt n_h Fabriken, die das Produkt herstellen. Der Preis, den der Kunde pro Einheit des Produkts bezahlt, ist dabei abhängig von der von allen Unternehmen produzierten Gesamtmenge des Produkts. Stellt nun jedes Unternehmen h die Menge $x^h = \sum_{i=1}^{n_h} x^{i,h}$ her, wobei $x^{i,h}$ die in Fabrik i produzierte Menge darstellt, ergibt sich eine mögliche Preisfunktion zu $P(x) = P_c - \sum_{h=1}^{H} x^h$ mit einem konstanten Term $P_c > 0$. Steigt die von allen Unternehmen produzierte Menge des Produkts, sinkt somit der Preis, den der Verbraucher bezahlt, und umgekehrt. Für jedes Unternehmen h fallen zusätzlich Produktionskosten der einzelnen Firmen i mit $K^h(x^h) = \sum_{i=1}^{n_h} K_i^h(x^{i,h})$ an, die von der entsprechend produzierten Menge abhängen. Die untereinander durch die Preisfunktion des Verbrauchers gekoppelten Gewinne ergeben sich folglich zu $F^h = P(x)x^h - K^h(x^h)$, die jedes Unternehmen eigensinnig optimieren will. Somit ergibt sich für Unternehmen $h = 1, ..., H$ das gekoppelte Optimierungsproblem

$$\min_{x^h} \sum_{i=1}^{n_h} K_i^h(x^{i,h}) - P(x)\left(\sum_{i=1}^{n_h} x^{i,h}\right). \tag{16}$$

Sollen darüber hinaus die Kostenfunktion K_i^h der einzelnen Fabriken privat bleiben, ist leicht zu erkennen, dass es sich bei diesem Problem um ein Multi-Cluster-Spiel handelt, bei dem die Unternehmen gegeneinander ein nichtkooperatives Spiel spielen und gleichzeitig versuchen, das kooperative Produktionsoptimum ihrer Firmen zu erreichen. Entsprechend soll ein nicht-kooperatives Nash-Gleichgewicht gefunden werden, welches die Optimalitätsbedingung (4) bzw. (5) erfüllt. Um die Anforderung der in Abschnitt 4 vorgestellten Algorithmen bezüglich der Spiel-Abbildung in Annahme 2 zu gewährleisten, werden quadratische Kostenfunktionen für die einzelnen Firmen gewählt, d. h. $K_i^h(x^{i,h}) = a_i^h(x^{i,h})^2 + b_i^h x^{i,h} + c_i^h$.

5.2 Setup

Zur Simulation wird ein Multi-Agenten-System erstellt, das aus insgesamt zehn verschiedenen Clustern besteht. Diese Cluster umfassen jeweils 11, 10, 9, 8, 7, 6, 5, 4, 3 und 2 Agenten, sodass sich die Gesamtagentenanzahl zu $n = 65$ ergibt. Jeder dieser Agenten repräsentiert eine Fabrik mit quadratischer Kostenfunktion, die Cluster stehen für die konkurrierenden Unternehmen. Um die beiden Algorithmen aus Abschnitt 4 bezüglich unterschiedlicher Kommunikationsstrukturen miteinander vergleichen zu können, werden zufällige, ungerichtete Graphen erstellt, die jeweils den Annahmen 3 bzw. 4 genügen. Den Fall bestmöglicher Kommunikation stellt ein voll vernetzter Graph dar, bei dem jeder Agent mit jedem anderen Agenten kommunizieren kann. Bezüglich der in den Algorithmen verwendeten Graphen bedeutet dies, dass die lokalen Graphen $\mathcal{G}^h$ innerhalb der Cluster h vollständig vernetzt sind und zusätzlich der globale Graph $\mathcal{G}$ in Algorithmus 2 alle Agenten miteinander vernetzt. Der Leader-Graph $\mathcal{G}^{\mathrm{L}}$ in Algorithmus 1 vernetzt entsprechend jeden Cluster-Leader mit jedem anderen Cluster-Leader. Im Folgenden wird dieses Setup als Szenario 1 bezeichnet. Für Szenario 2 werden spärliche, zufällige Graphen verwendet. Die lokalen Graphen $\mathcal{G}^h$ enthalten genauso viele Kanten wie Agenten, mit Ausnahme des Clusters mit nur zwei Agenten, der mit einer einzigen Kante bereits voll vernetzt ist. Dasselbe gilt für den Leader-Graph $\mathcal{G}^{\mathrm{L}}$ für Algorithmus 1, während der globale Graph $\mathcal{G}$ für Algorithmus 2 mit insgesamt 75 Kanten bei 65 Knoten vernetzt ist. Für Szenario 3 schließlich werden die lokalen Graphen $\mathcal{G}^h$ und der Leader-Graph $\mathcal{G}^{\mathrm{L}}$ aus Szenario 2 übernommen und der globale Graph $\mathcal{G}$ so angepasst, dass dieser aus einer Kombination aus lokalen Graphen und dem Leader-Graph bestehen, sodass die Kommunikationsverbindungen für Algorithmus 2 exakt mit denjenigen übereinstimmen, die in Algorithmus 1 verwendet werden. Die Wahl der Szenarien hat folgenden Hintergrund: In Szenario 1 kann durch die voll vernetzten Graphen davon ausgegangen werden, dass die Kommunikation kein zusätzliches Nadelöhr für die Konvergenz darstellt. Nur die Inter-Cluster-Kommunikation ist hier für

Tab. 1: Übersicht über Anzahl der ungerichteten Kommunikationsverbindungen der Graphen für die drei Szenarien, wobei $\Delta_{n-1} = n(n-1)/2$ der Dreieckszahl entspricht. $\mathcal{G}^h$: Lokale Graphen in Cluster h, $\mathcal{G}^L$: Leader-Graph von Algorithmus 1, $\mathcal{G}$: Globaler Graph Algorithmus 2.

	$\mathcal{G}^h$	$\mathcal{G}^L$	$\mathcal{G}$
Szen. 1	Δ_{n_h-1}	45	2080
Szen. 2	n_h	10	75
Szen. 3	n_h	10	74

Tab. 2: Schrittweiten und Iterationsanzahlen der Agenten, bis alle Schätzungen eine relative Genauigkeit von 10^{-4} zum Optimum besitzen.

		α	Iterationen
Szen. 1	Alg. 1	0,027	5773
	Alg. 2	0,088	2411
Szen. 2	Alg. 1	0,033	5560
	Alg. 2	0,083	3145
Szen. 3	Alg. 2	0,114	10168

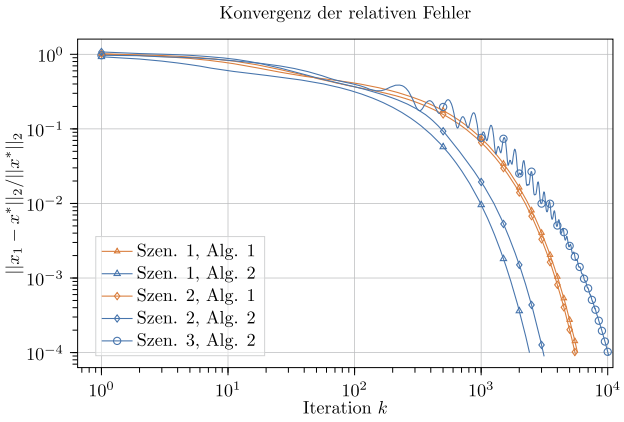

Abb. 4: Vergleich der Konvergenzgeschwindigkeiten der relativen Fehler für Algorithmus 1 und 2 bezüglich Nash-Cournot-Spiel unter verschiedenen Szenarien und Kommunikationsarchitekturen.

Algorithmus 1 durch die Beschränkung auf den Leader-Graphen im Vergleich zu Algorithmus 2 eingeschränkt. In Szenario 2 wird die Gesamtanzahl der Kommunikationsverbindungen wie beschrieben reduziert. Dadurch soll die Bedeutung der Anzahl der Graphverbindungen auf die Konvergenzgeschwindigkeit deutlich gemacht werden. In Szenario 3 verwenden schließlich beide Algorithmen die exakt gleiche Kommunikationsstruktur, im Fall von Algorithmus 1 entspricht diese dem Szenario 2. Hier soll untersucht werden, welcher Algorithmus bei Einschränkung auf ausschließliche Leader-Kommunikation eine schnellere Konvergenz aufweist. Die Graphstrukturen sind in Tabelle 1 anhand ihrer Kenngrößen zusammengefasst.

5.3 Ergebnisse

Die Schrittweiten α der beiden Algorithmen werden unter Verwendung einer Rastersuche so gewählt, dass sich für den jeweiligen Graphen und Algorithmus die geringste Anzahl an Iterationen bis zu einer Genauigkeit von 10^{-3} zum tatsächlichen Optimum einstellt. Die Ergebnisse sind in Tabelle 2 aufgeführt. Im untersuchten Intervall zeigt sich, dass die Verfahren ab einer bestimmten Schrittweite instabil werden und auch für höhere Schrittweiten nicht mehr konvergieren. In den Untersuchungen liegt die optimale Schrittweite in der Nähe dieses Grenzfalls. Das Optimum x^* des Spiels wird im Vorfeld durch das nicht-verteilte Diagonalisierungsverfahren nach Gauß-Seidel, siehe z. B. [3], ermittelt. In Abbildung 4 sind die Verläufe der relativen Fehler der Schätzungen des ersten Agenten, d. h. $\delta_1^{\text{rel}} = ||x_1 - x^*||_2/||x^*||_2$, dargestellt. Die Simulationen werden abgebrochen, sobald alle Agenten relative Fehler von $\delta_i^{\text{rel}} \leq 10^{-4}$, $\forall i = 1, \ldots, 65$, unterschritten haben. Somit ist die Konsensusbedingung zu einer ausreichenden Genauigkeit ebenfalls erfüllt. Während die Farben in der Grafik zwischen den Algorithmen 1 und 2 unterscheiden, stehen die Marker für die verschiedenen Test-Szenarien: Dreieck für Szenario 1, Raute für Szenario 2 und Kreis für Szenario 3. Da sich die Graphen von Szenario 2 und 3 für den Algorithmus 1 nicht unterscheiden, wird hier nur der Verlauf für diesen Algorithmus für das zweite Szenario geplottet. Die Iterationsanzahl ist zusätzlich in Tabelle 2 aufgeführt.

Zunächst werden die Resultate für Algorithmus 1 betrachtet. Die für diesen Algorithmus relevanten Szenarien 1 und 2 der voll und spärlich vernetzten Graphen erzeugen sehr ähnliche Konvergenzverläufe, wobei die Iterationsanzahl für Szenario 2 mit 5560 Iterationen 213 Iterationen unterhalb der von Szenario 1 liegt. Dieses Ergebnis ist kontraintuitiv, wenn davon ausgegangen wird, dass bessere Vernetzungen kürzere Konvergenzzeiten nach sich ziehen.

Im Fall von Algorithmus 2 stellt sich das insgesamt beste Ergebnis bei einem voll vernetzten Graphen mit 2411 Iterationen ein, das zweitbeste Ergebnis mit 3145 für Szenario 2 und schließlich das schlechteste Ergebnis mit 10168 Iterationen in Szenario 3. Die Ergebnisse für Szenario 1 und 2 entsprechen der Erwartung, dass sich bei höherer Anzahl der Kommunikationsverbindungen eine schnellere Konvergenz einstellen sollte. Bei einem Blick in Tabelle 1 fällt die wesentlich höhere Anzahl der Kommunikationsverbindungen bei den voll vernetzten Graphen

des Szenarios 1 für die Aktionsaktualisierung des Algorithmus 2 im Vergleich zu Algorithmus 1 auf: Während Algorithmus 1 im Aktualisierungsschritt der Aktionen insgesamt über 265 Kanäle kommuniziert, verschickt Algorithmus 2 über insgesamt 2080 Verbindungen Informationen. Diese niedrigere Anzahl bei Algorithmus 1 ist durch die in Abschnitt 4 diskutierte Einschränkung der intercluster Kommunikation auf den ausschließlichen Leader-Austausch zurückzuführen. Offensichtlich führt diese Einschränkung sowohl im Szenario 1 als auch im Szenario 2 zu einer geringeren Anzahl der Kommunikationsschritte und daraus folgend auch zu deutlich höheren Iterationanzahlen bis zum annähernden Erreichen des Optimums.

Für Szenario 3 allerdings zeigt sich eine deutliche Verschlechterung der Konvergenzzeit von Algorithmus 2. Zur Erinnerung: Hier besteht der globale Kommunikationsgraph $\mathcal{G}$ aus der Vereinigung der lokalen Graphen $\mathcal{G}^h$ und des Leader-Graphen $\mathcal{G}^{\mathrm{L}}$, wodurch die benutzbaren Kantenmengen für die Aktualisierung der Aktionsschätzungen bei Algorithmus 1 und 2 nun gleich sind. Bei Vergleich der Konvergenzzeiten von Algorithmus 1 und 2 zeigt sich, dass Algorithmus 1 mit 5560 eine deutlich geringe Iterationsanzahl bis zum Zielfehler aufweist, obwohl Algorithmus 2 in Szenario 1 und 2 bessere Ergebnisse erzielen kann. Zusätzlich treten Schwingungen in der Fehlerkonvergenz des Algorithmus 2 auf. Erstaunlich ist dieses Ergebnis außerdem, da sich die Kantenanzahl zwischen Szenario 2 und 3 für den globalen Graphen $\mathcal{G}$ kaum verändert hat (siehe Tabelle 1). Daraus lässt sich schließen, dass für die Konvergenzgeschwindigkeit nicht nur die reine Anzahl der Kommunikationskanäle relevant ist; eine scheinbar ebenso wichtige Rolle spielt, ob die Verbindungen unabhängig von Clusterzugehörigkeit gesetzt werden, oder, wie im Szenario 3 der Fall, die intercluster Kommunikation stark eingeschränkt ist. Nun stellt sich die Frage, warum Algorithmus 1 offensichtlich deutlich besser mit dieser Schwierigkeit in Szenario 3 umgehen kann. Die Antwort liegt in der Gewichtung der Inter- und Intra-Cluster-Kommunikatin zueinander. Bei einem Blick auf die Aktualisierungsgleichung (14a) eines Cluster-Leaders des Algorithmus 1 lässt sich feststellen, dass die Aktualisierung der eigenen Aktion gleich stark von den Aktionen der cluster-fremden Agenten und der Agenten des eigenen Clusters beeinflusst werden, bedingt durch den Faktor von 1/2 vor den beiden gewichteten Summen. Damit haben die Aktionsschätzungen der Agenten anderer Cluster einen stärkeren Einfluss auf die Aktualisierung als es in Algorithmus 2 der Fall ist. Hier sind die in der Simulation verwendeten Gewichtungen etwa gleichmäßig über die eingehenden Informationen der Nachbarn verteilt. Das bedeutet, dass, wenn innerhalb der Cluster viele Kommunikationsverbindungen existieren, der Einfluss der Inter-Cluster-Kommunikation auf die Aktualisierung sinkt. Und da in Szenario 3 ausschließlich die Cluster-Leader sich über die Clustergrenzen hinweg austauschen können, sinkt der Einfluss im Vergleich zum allgemeineren Fall noch weiter. Eine Anpassung der Gewichtung könnte das Ergebnis für Algorithmus 2 verbessern.

Zusammenfassend lässt sich festhalten, dass Algorithmus 2 durch geringere Einschränkungen der Inter-Cluster-Kommunikation bei höherer Anzahl der Kommunikationskanäle deutlich weniger Iterationen bis zu einer Genauigkeit von 10^{-4} zum Optimum benötigt als Algorithmus 1. Damit konnten die Überlegungen aus dem Abschnitt 4 bestätigt werden. Im Falle von geringer Kommunikation zwischen den Clustern bei gleichzeitig hoher Kommunikation innerhalb der Cluster kann Algorithmus 1 durch eine Gleichgewichtung von Inter- zu Intra-Cluster-Kommunikation bei den Leader-Agenten schneller das Optimum erreichen. Damit wird deutlich, dass in der betrachteten Simulation nicht nur die reine Anzahl der Kommunikationsverbindungen für die Konvergenz eine Rolle spielt, sondern auch die Verteilung der Kanten auf Inter- und Intra-Cluster-Kommunikation.

6 Fazit

In diesem Artikel wird die Bedeutung des Gradient-Tracking-Verfahrens in Multi-Cluster-Spielen herausgearbeitet. Als Ergebnis der Simulationen lässt sich festhalten, dass hierarchische Clusterstrukturen, in denen Agenten in Leader und Follower eingeteilt sind, die Kommunikation im Vergleich zu nicht-hierarchischen Strukturen so stark einschränken, dass dies einen erheblichen Einfluss auf die Konvergenzgeschwindigkeit hat. Allerdings hat der untersuchte Leader-Follower Algorithmus durch die besondere Gewichtung seiner Kommunikation Vorteile, wenn nur ein Agent pro Cluster mit Agenten anderer Cluster kommuniziert. Inwieweit sich diese Erkenntnis verallgemeinern lässt ist, kann Gegenstand zukünftiger Forschung sein.

Finanzierung: Diese Arbeit wurde durch die Deutsche Forschungsgemeinschaft gefördert - Schwerpunktprogramm 1984.

Literatur

1. Ahat, M., S.B. Amor, M. Bui, A. Bui, G. Guérard and C. Petermann. 2013. Smart Grid and Optimization. *American Journal of Operations Research* 3(1): 196–206.

2. Dall'Anese, E., H. Zhu and G.B. Giannakis. 2013. Distributed optimal power flow for smart microgrids. *IEEE Transactions on Smart Grid* 4(3): 1464–1475.
3. Kanzow, C. and A. Schwartz. 2018. *Spieltheorie: Theorie und Verfahren zur Lösung von Nash- und verallgemeinerten Nash-Gleichgewichtsproblemen*. Birkhäuser.
4. Kasbekar, G.S. and S. Sarkar. 2012. Pricing games among interconnected microgrids. In: *IEEE Power and Energy Society General Meeting*.
5. Meng, M. and X. Li. 2020. On the linear convergence of distributed Nash equilibrium seeking for multi-cluster games under partial-decision information. arXiv: 2005.06923.
6. Mengelkamp, E., J. Gärttner, K. Rock, S. Kessler, L. Orsini and C. Weinhardt. 2018. Designing microgrid energy markets: A case study: The Brooklyn Microgrid. *Applied Energy* 210: 870–880.
7. Nedić, A., A. Olshevsky and W. Shi. 2017. Achieving geometric convergence for distributed optimization over time-varying graphs. *SIAM* 27(4): 2597–2633.
8. Pang, Y. and G. Hu. 2021. Gradient-free Nash equilibrium seeking in *n*-cluster games with uncoordinated constant step-sizes. arXiv: 2008.13088.
9. Pu, S., W. Shi, J. Xu and A. Nedić. 2021. Push-Pull Gradient Methods for Distributed Optimization in Networks. *IEEE Transactions on Automatic Control* 66(1): 1–16.
10. Qu, G. and Na Li. 2018. Harnessing smoothness to accelerate distributed optimization. *IEEE Transactions on Control of Network Systems* 5(3): 1245–1260.
11. Tatarenko, T. and A. Nedić. 2020. Geometric convergence of distributed gradient play in games with unconstrained action sets. *IFAC-PapersOnLine* 53: 3367–3372, Elsevier B. V.
12. Tatarenko, T., J. Zimmermann and J. Adamy. 2021. Gradient play in *n*-cluster games with zero-order information. arXiv: 2107.12648.
13. Xin, R., C. Xi and U.A. Khan. 2019. FROST – Fast row-stochastic optimization with uncoordinated step-sizes. *Eurasip Journal on Advances in Signal Processing* 2019(1): 1.
14. Ye, M. and G. Hu. 2017. Simultaneous social cost minimization and Nash equilibrium seeking in non-cooperative games. In: *Chinese Control Conference, CCC*, pp. 3052–3059.
15. Ye, M., G. Hu and F.L. Lewis. 2018. Nash equilibrium seeking for N-coalition noncooperative games. *Automatica* 95: 266–272.
16. Ye, M., G. Hu, F.L. Lewis and L. Xie. 2017. A unified strategy for solution seeking in graphical N-coalition noncooperative games. *IEEE Transactions on Automatic Control* 64(11): 4645–4652.
17. Ye, M., G. Hu and S Xu. 2020. An extremum seeking-based approach for Nash equilibrium seeking in N-cluster noncooperative games. *Automatica* 114: 108815.
18. Zeng, X., J. Chen, S. Liang and Y. Hong. 2019. Generalized Nash equilibrium seeking strategy for distributed nonsmooth multi-cluster game. *Automatica* 103: 20–26.
19. Zimmermann, J., T. Tatarenko, V. Willert and J. Adamy. 2021. Solving leaderless multi-cluster games over directed graphs. *European Journal of Control* 62: 14–21.
20. Zou, Y., B. Huang, Z. Meng and W. Ren. 2019. Distributed Nash equilibrium seeking algorithms for two-layer constrained non-cooperative games. In: *Chinese Control Conference, CCC*, pp. 5776–5781.

Autoreninformationen

Jan Zimmermann
Technische Universität Darmstadt,
Darmstadt, Deutschland
jan.zimmermann@rmr.tu-darmstadt.de

M. Sc. Jan Zimmermann ist wisschenschaftlicher Mitarbeiter am Fachgebiet Regelungsmethoden und Robotik unter der Leitung von Prof. Dr.-Ing. J. Adamy am Institut für Automatisierungstechnik und Mechatronik im Fachbereich Elektrotechnik und Informationstechnik der Technischen Universität Darmstadt. Hauptarbeitsgebiete: Verteilte Optimierung, Spieltheorie.

Tatiana Tatarenko
Technische Universität Darmstadt,
Darmstadt, Deutschland
tatiana.tatarenko@rmr.tu-darmstadt.de

Dr. rer. nat. Tatiana Tatarenko ist Gruppenleiterin der Forschergruppe Verteilte Optimierung und Spieltheorie des Fachgebietes Regelungsmethoden und Robotik unter der Leitung von Prof. Dr.-Ing. J. Adamy am Institut für Automatisierungstechnik und Mechatronik im Fachbereich Elektrotechnik und Informationstechnik der Technischen Universität Darmstadt. Hauptarbeitsgebiete: Angewandete Mathematik, verteilte Optimierung, Spieltheorie.

Volker Willert
Hochschule für angewandte Wissenschaften Würzburg-Schweinfurt,
Schweinfurt, Deutschland
volker.willert@fhws.de

Prof. Dr.-Ing. Volker Willert ist Professor für maschinelles Sehen der Fakultät Elektrotechnik der Hochschule für angewandte Wissenschaften Würzburg-Schweinfurt. Hauptarbeitsgebiete: Maschinelles Sehen, Mobile Robotik, Maschinelles Lernen, Multi-Agenten-Systeme.

Jürgen Adamy
Technische Universität Darmstadt,
Darmstadt, Deutschland
adamy@rmr.tu-darmstadt.de

Prof. Dr.-Ing. Jürgen Adamy ist Professor des Fachgebietes Regelungsmethoden und Robotik am Institut für Automatisierungstechnik und Mechatronik im Fachbereich Elektrotechnik und Informationstechnik der Technischen Universität Darmstadt. Hauptarbeitsgebiete: Nichtlineare Regelungstechnik, Computational Intelligence, Mobile Robotik.