Autonomous Agents and Multi-Agent Systems (2020) 34:18
https://doi.org/10.1007/s10458-019-09440-y

Fast core pricing algorithms for path auction

Hao Cheng[1] · Wentao Zhang[1] · Yi Zhang[1] · Lei Zhang[1] · Jun Wu[1] · Chongjun Wang[1]

Published online: 14 January 2020

Abstract
Path auction is held in a graph, where each edge stands for a commodity and the weight of this edge represents the prime cost. Bidders own some edges and make bids for their edges. The auctioneer needs to purchase a sequence of edges to form a path between two specific vertices. Path auction can be considered as a kind of combinatorial reverse auctions. Core-selecting mechanism is a prevalent mechanism for combinatorial auction. However, pricing in core-selecting combinatorial auction is computationally expensive, one important reason is the exponential core constraints. The same is true of path auction. To solve this computation problem, we simplify the constraint set and get the optimal set with only polynomial constraints in this paper. Based on our constraint set, we put forward two fast core pricing algorithms for the computation of bidder-Pareto-optimal core outcome. Among all the algorithms, our new algorithms have remarkable runtime performance. Finally, we validate our algorithms on real-world datasets and obtain excellent results.

Keywords Path auction · Core · Pricing algorithm · Constraint set

Part of the contents in this paper were published in AAMAS '18 [6].

✉ Hao Cheng
chengh@smail.nju.edu.cn

Wentao Zhang
zhangwent963@gmail.com

Yi Zhang
njuzhangyi@smail.nju.edu.cn

Lei Zhang
zhangl@nju.edu.cn

Jun Wu
wujun@nju.edu.cn

Chongjun Wang
chjwang@nju.edu.cn

1 National Key Laboratory for Novel Software Technology, Department of Computer Science and Technology, Nanjing University, Nanjing, China

1 Introduction

Path auction has been studied extensively [13, 27, 31] since Nisan and Ronen [26] introduced algorithmic mechanism design. In path auction, auctioneer tries to buy a path between two given vertices while the edges in the graph are owned by some bidders. The cost of each edge is the private information of its owner. There are many application scenarios of path auction such as transport routing, power transmission and so on.

The classic mechanism for path auction is the well-known Vickrey–Clark–Groves (VCG) mechanism, where bidders pay the externalities they impose on all the other bidders. VCG mechanism is the unique mechanism that can guarantee efficient allocation and incentive compatibility theoretically. However, VCG mechanism has some issues. On one hand, it may result in a low revenue to the auctioneer [2]. On the other hand, VCG mechanism may be manipulated by some false-name bids [30]. These issues have led to considerable interests in core-selecting mechanism [10, 11].

Core-selecting mechanism has been studied in the area of combinatorial auctions [7, 24, 28, 30]. This mechanism is false-name-proof and has better revenue performance than VCG mechanism so that it is widely used in auctions such as spectrum auctions [8], procurement auctions [32] and TV advertising auctions [11]. Core-selecting mechanism selects the outcome from the core so that no coalition in the auction can improve upon the outcome. However, core-selecting mechanism has two main computation problems. One is the winner determination problem [28]. In general combinatorial auction, winner determination is to find an efficient allocation. It can be reduced to an integer programming problem, which is NP-hard. The other problem is the exponential core constraints. The result of core-selecting mechanism is a polytope, composed of numerous constraints. Each constraint is relevant to a coalition and the total number of the possible coalitions for a game of n players is $2^n - 1$, which is exponential. Some heuristic algorithms were proposed to solve this problem. Core Constraint Generation algorithm (CCG) is a prevalent algorithm in practice [12], which only considers the most valuable core constraints to get the bidder-Pareto-optimal core outcome. It reduces the coalitions to a moderate number in expectation. However, its performance isn't good enough for some online applications and it lacks a theoretical guarantee for the performance.

In path auction, winner determination problem is equivalent to the problem of computing the shortest path. It is easy to compute through some existing algorithms such as Dijkstra algorithm. However, the number of core constraints is still exponential, which is the key problem to solve. In this paper, we tackle this problem by investigating the structural properties of core constraints to and gain excellent results theoretically and practically. Our contributions are stated as follows:

1. We simplify the original exponential constraint set ($C1$) to a polynomial constraint set ($C2$) without redundant constraints.
2. We provide a new convenient format of constraint set ($C3$). Based on the three constraint sets, we propose three direct pricing algorithms to compute the bidder-Pareto-optimal core outcome.
3. We also put forward a novel fast algorithm called Bellman–Ford Path Auction (BFPA) algorithm. This algorithm only needs to run the single-source shortest path algorithm once and we give the proof for its correctness.
4. We validate our approaches on real-world datasets and obtain excellent results.

The remainder of the paper is organized as follows. We begin by discussing related work of core-selecting mechanism and path auction. Section 2 describes the background of core-selecting path mechanism. Section 3 mainly presents a proof of the equivalence $(C2) \Leftrightarrow (C1)$ while Sect. 4 gives a proof of $(C3) \Leftrightarrow (C2)$. After the content of constraint sets, Sect. 5 demonstrates four core pricing algorithms that are LPPA-C1, LPPA-C2, LPPA-C3 and CCG-VCG. Then in Sect. 6, we propose a new fast algorithm called BFPA algorithm and prove its correctness. Section 7 presents the results of our experiments. Section 8 concludes with a summary of what we have accomplished and a discussion of future work.

1.1 Related work

In combinatorial auctions, VCG mechanism is a prevalent mechanism in the field of auction [7, 18, 29]. However, it is rarely used in practice due to some issues, among which one is the potentially low revenue [2, 3]. In [2], ascending proxy auction was proposed to resolve this issue. This ascending auction uses a bidder-Pareto-optimal core outcome as the final payment. Then, the research of core-selecting mechanism was further developed by [9–12]. Computing core outcome is at least as hard as a particular winner determination problem, which is effectively a separation oracle for the core polytope by inputting the truncated values [11]. The Ellipsoid algorithm in [17] can therefore be used to solve this problem, but this is rather slow in practice. Various heuristic algorithms have been put forward to address this problem. In [12], Core Constraint Generation algorithm (CCG) was proposed as a fast algorithm to obtain a specific core outcome in combinatorial auction. This algorithm was then developed to be more practical in [4, 11, 15]. Another approach is using the recent fast cutting-plane methods [23] to obtain a payment result outcome with an ε-approximation with high probability. In the specific scene of rich advertising auctions [19], an approximate algorithm was proposed to find a bidder-Pareto-optimal core outcome with almost linear number of calls to the welfare maximization oracle.

In addition to the literature mentioned above, our work is also related to the literature on path auction. The problem of designing economic mechanisms for path auction was first studied in [26], where VCG mechanism is applied to find the shortest path. It is shown that the VCG payments can be computed using $|V|$ runs of Dijkstra algorithm in $O(|E||V| + |V|^2 \log |V|)$ time. It is later proved that if the underlying graph is undirected, the VCG payments can be computed in only $O(|E| + |V| \log |V|)$ time [20]. Previous work has found that VCG path mechanism can be forced to make arbitrarily high overpayment in the worst case. In fact the result can be generalized to include all truthful path mechanisms [14, 22]. This led to the study of frugal path mechanisms [1]. Previous work has also studied the VCG overpayment in the internet inter-domain routing graph [16] and large random graphs [21]. Then core-selecting path mechanism was designed by [31] as a frugal path mechanism. They put forward a new constraint set for the core polytope with 2^n core constraints, where n is the network diameter. However, they didn't offer an efficient algorithm. In this paper, we propose several deterministic polynomial pricing algorithms for path auction, which also have theoretical guarantee for the performance.

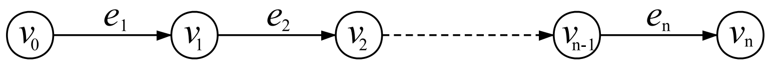

Fig. 1 The winner path $P_w(v_0, v_n)$

2 Preliminaries

In our model, the social network is represented by a directed weighted graph $G = (V, E)$. The edges represent the commodities in the auction, owned by strategic agents. Each edge e_i has a prime cost $c_{e_i} \in R^+$, which is the private information of the agent who owns this edge. In this paper, we assume that each agent only owns one edge e_i so that e_i can be used to represent the corresponding agent. Each agent is also a bidder and makes a bid $b_{e_i} > 0$ for his edge. The auctioneer aims to buy an edge set to form a walk from the source vertex v_0 to the target vertex v_n. We denote this walk by

$$W_G(v_0, v_n) = v_0 e_1 v_1 e_2 \dots v_{n-1} e_n v_n$$

$W_G(v_0, v_n)$ is a finite non-empty sequence with alternate vertices and edges, representing a walk from v_0 to v_n in the graph G. $e_i = (v_{i-1}, v_i)$ represents the edge from v_{i-1} to v_i. There may be repeated vertices in the walk $W_G(v_0, v_n)$. However, it is costly to buy a walk that passes a vertex v_i repeatedly in the auction. Thus, we focus on the paths that don't include repetitive vertices. We use $P_w(v_0, v_n)$ to represent the path bought by the auctioneer in the auction, known as the winner path. As shown in Fig. 1, winner path can be expressed as

$$P_w(v_0, v_n) = v_0 e_1 v_1 e_2 \dots v_{n-1} e_n v_n$$

where n is the number of winners. For $P_w(v_0, v_n)$, its vertex set $\{v_0, v_1, \dots, v_n\}$ is represented by $V_w(v_0, v_n)$ (abbreviated as V_w for convenience), and its edge set $\{e_1, e_2, \dots, e_n\}$ is represented by $E_w(v_0, v_n)$ (abbreviated as E_w). E_w is just the winner set in this auction. The auctioneer will pay for these edges, which generates a payment set as $P = \{p_{e_1}, p_{e_2}, \cdots, p_{e_n}\}$. This becomes a problem of mechanism design. The winner set E_w and the payment set P are the outcome of path auction. We assume that the agents have quasi-linear utility function in this paper. Denote by π_{e_i} the utility of bidder e_i and it is defined as follows.

$$\pi_{e_i} = \begin{cases} p_{e_i} - c_{e_i} & e_i \in E_w \\ 0 & e_i \notin E_w \end{cases} \tag{1}$$

Denote the auctioneer by 0, then

$$\pi_0 = -\sum_{e_i \in E_w} p_{e_i} \tag{2}$$

Denote by Π the utility of the system including bidders and auctioneer (i.e., social welfare) and we have

$$\Pi = \sum_{e_i \in E_w} \pi_{e_i} + \pi_0 = -\sum_{e_i \in E_w} c_{e_i} \tag{3}$$

We use $P_G(v_0, v_n)$ to represent the shortest path from v_0 to v_n in graph G. The edge set and vertex set of this path are also represented by $V_G(v_0, v_n)$ and $E_G(v_0, v_n)$ respectively, and

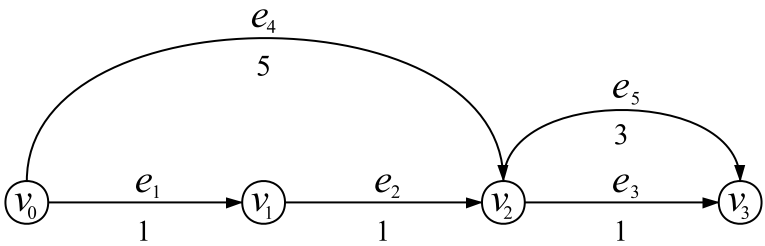

Fig. 2 An example of path auction. There are 5 bidders e_1, e_2, e_3, e_4, e_5 with cost 1, 1, 1, 5, 3

the cost is represented by $d_G(v_0, v_n)$. According to Eq. (3), the maximum social welfare is $-d_G(v_0, v_n)$ when the shortest path is selected as the winner path.

In general auctions, bidders are considered to be rational and strategic. In order to improve their utilities, they will take some strategies such as misreporting costs, registering some fake accounts to bid, forming coalitions with other bidders and so on. Thus, the designed mechanism is expected to defend against these manipulation strategies. Here are the definitions of three expected properties in the auction mechanism.

Definition 1 *(Individual Rational)* A mechanism is individual rational if and only if each bidder is guaranteed a non-negative utility.

In path auction, individual rational means the outcome of the mechanism needs to satisfy IR constraints below.

$$\pi_{e_i} \geq 0 \quad \forall e_i \in E$$

Definition 2 *(Efficient)* A mechanism is efficient if and only if the outcome of this mechanism gets the maximum social welfare.

According to Eq. (3), it means that the mechanism should select the edges on the shortest path as the winners.

Definition 3 *(Incentive Compatible)* A mechanism is incentive compatible if and only if reporting the true cost is each bidder's dominant strategy in this mechanism.

Well-known VCG mechanism is the unique mechanism that satisfies these three properties theoretically, but it has some issues which will be discussed in Sect. 2.1.

2.1 VCG mechanism

In VCG mechanism, the allocation rule is to choose the shortest path as the winner path, and for each winner e_i, the VCG payment is

$$p_{e_i} = d_{G \backslash e_i}(v_0, v_n) - d_G(v_0, v_n) + c_{e_i} \tag{4}$$

where $G \backslash e_i$ is the graph removing e_i from G. In VCG mechanism, the bidder is truthful so that his bid price is his true value. For instance, in Fig. 2, the shortest path $P_G(v_0, v_n)$ is

$v_0 e_1 v_1 e_2 v_2 e_3 v_3$. The bidders e_1, e_2, e_3 win and they obtain the payment of 4, 4, 3 respectively. Therefore, the auctioneer needs to pay 11 and the final revenue is −11.

We can see that this result produces a low revenue for the auctioneer while the optimal revenue should be −3. VCG mechanism causes an overpayment for the auctioneer. Besides, VCG mechanism is vulnerable to false-name manipulation. assuming that e_1 and e_2 are two fake accounts of one agent e_6, this agent owns an edge from v_0 to v_2, with a cost of 2. If e_6 bids truthfully, he will get a payment of 5 in VCG mechanism, but if e_6 uses the two fake accounts e_1, e_2, he will get a payment of 8 totally. This could form a false-name manipulation in the auction and VCG mechanism couldn't defense this manipulation.

Due to these two issues of VCG mechanism, the study of frugal and false-name-proof auction mechanism was initiated. In these studies, core-selecting auction mechanism is a well-known combinatorial auction mechanism, which has been adopted in spectrum auction [9, 11].

2.2 Core-selecting path mechanism

In the case of path auction, core-selecting mechanism is described as follows.

Model the path auction as a cooperative game (N, W) and use the core as a solution concept. N represents all the players in this game. Note that $N = G \cup \{0\}$ where all bidders are included in graph G. W represents the social welfare. Denote by L a subset of N and its welfare is defined as

$$W(L) = \begin{cases} -d_{L\setminus\{0\}}(v_0, v_n), & 0 \in L \\ 0, & 0 \notin L \end{cases} \tag{5}$$

Definition 4 *(Core outcome)* In path auction, a core outcome is an allocation and payment profile such that the utility profile $\pi = \{\pi_{e_1}, \pi_{e_2}, \dots, \pi_{e_n}\}$ satisfies

$$\sum_{e_i \in L} \pi_{e_i} \geq W(L), \forall L \subset N \tag{6}$$

Given that $L = N$ in (6), we obtain

$$\sum_{e_i \in N} \pi_{e_i} \geq W(N) \tag{7}$$

where $W(N) = -d_G(v_0, v_n)$. Besides, $-d_G(v_0, v_n)$ is also the maximum value of the social welfare, then we have the following equation.

$$\sum_{e_i \in N} \pi_{e_i} = W(N) \tag{8}$$

Therefore, in core-selecting auction mechanism, the shortest path should be selected as the winner path (i.e. $P_w(v_0, v_n) = P_G(v_0, v_n)$). Assuming that $L = \{e_i\}$, we obtain

$$\pi_{e_i} \geq 0 \tag{9}$$

This is just one IR constraint. Thus, the properties of efficient and individual rational are included in (6).

Generally speaking, inequality (6) means that the welfare of the subset L under the profile π is not lower than that under the definition of formula (5). This constraint prevents the agents of L from building a coalition because they couldn't get higher profit than current result in this way. Core is defined as the total set of core outcomes. Next, we can define the core-selecting path mechanism.

Definition 5 *(Core-selecting path mechanism)* A path auction mechanism is core-selecting if

1) it selects the shortest path as the winner path;
2) the payment set P is computed so that $P \in$ core.

Core-selecting path mechanism is individual rational and efficient, and it satisfies the core property in the case of bidders reporting their cost truthfully [31]. The core property means no coalition can form a mutually beneficial renegotiation among themselves. However, core-selecting path mechanism relaxes the property of incentive compatible so that bidders may not report their true costs. To tackle this problem, we provide a theorem about the bidders' misreporting as follows.

Theorem 1 *In core-selecting path mechanism, given that the other bidders bid truthfully, the bidder e_i's maximum misreporting price is his VCG price.*

Proof Assuming that e_i bids more than his VCG price, then the shortest path will be changed into another path that doesn't include e_i. Core-selecting path mechanism is efficient and always chooses the shortest path so that e_i would lose. Thus, VCG price is an upper bound for misreporting. □

This problem also leads to some researches of payment rules in core-selecting mechanism [11]. But in this paper, we focus on the computation problem of core-selecting path auction. Therefore, we make the assumption that bidders report their costs truthfully (*i.e.* $b_{e_i} = c_{e_i}$) in the following discussion.

2.3 Core constraints

To get a core outcome of core-selecting path mechanism, the first step is to find the shortest path in the graph, which is simple. The next step is to generate the constraints in (6), but the number of constraints is too huge to compute. Fortunately, it could be simplified and has been simplified into ($C1$) in [31].

$$(C1):\sum_{e_i\in x} p_{e_i} \le d_{G\backslash x}(v_0, v_n) - \left(d_G(v_0, v_n) - \sum_{e_i\in x} c_{e_i}\right), \forall x \subset E_w \tag{10}$$

In ($C1$), x is a nonempty subset of E_w. $G\backslash x$ is the graph removing the edges in x from G. $d_{G\backslash x}(v_0, v_n)$ represents the cost of the shortest path from v_0 to v_n in $G\backslash x$ and $d_G(v_0, v_n, G) - \sum_{e_i\in x} c_{e_i}$ is the total cost of the edge set $E_w\backslash x$.

We assume that each edge in E_w isn't a cut edge for the connectivity from v_0 to v_n. A cut edge means there exists no path from v_0 to v_n after removing this edge. Cut edge will form a monopoly, which is not allowed in core-selecting auction mechanism. Notice that this assumption is different from that in [6]. Therefore, the path $P_{G\backslash x}(v_0, v_n)$ may not exist in the graph $G\backslash x$ and we won't consider the corresponding constraint in $(C1)$ in this case.

In Fig. 2, E_w is the edge set $\{e_1, e_2, e_3\}$ and the total cost of $P_G(v_0, v_n)$ is 3. According to $(C1)$, we need to find the nonempty subsets of $\{e_1, e_2, e_3\}$ and them into the formula (10). Then we obtain 7 constraints as

$$\begin{cases} p_{e_1} \leq 4, p_{e_2} \leq 4, p_{e_3} \leq 3 \\ p_{e_1} + p_{e_2} \leq 5, p_{e_2} + p_{e_3} \leq 7, p_{e_1} + p_{e_3} \leq 7 \\ p_{e_1} + p_{e_2} + p_{e_3} \leq 8 \end{cases} \tag{11}$$

For example, given that $x = \{e_1\}$, the cost of the shortest path is 6 in the graph $G\backslash\{e_1\}$, so we have the constraint $p_{e_1} \leq 6 - (3 - 1) = 4$. Similarly, we can get other constraints. Moreover, we also need to consider the following three IR constraints.

$$p_{e_1} \geq 1, p_{e_2} \geq 1, p_{e_3} \geq 1 \tag{12}$$

There are 10 constraints above and an outcome satisfying all the constraints will be a core outcome ($p_{e_1} = 2$, $p_{e_2} = 1$, $p_{e_3} = 2$ for instance). All these constraints form a problem of linear programming and the core is the feasible region of this linear programming. However, the number of constraints is $2^n - 1$ in the worst case, where n is the number of winners. In order to generate the constraint relevant to x, $d_{G\backslash x}(v_0, v_n)$ needs to be calculated through the shortest path algorithm. Thus, $2^n - 1$ constraints in $(C1)$ indicate that the shortest path algorithm needs to be run exponential times, which is time-consuming. To reduce the computational complexity, we put forward a new constraint set $(C2)$ in this paper.

3 Optimal constraint set (C 2)

We denote by $P_w(v_i, v_j)$ the subpath of $P_w(v_0, v_n)$ from v_i to v_j. Similarly, the vertex set and edge set of this subpath are denoted by $V_w(v_i, v_j)$ and $E_w(v_i, v_j)$ respectively. Then constraint set $(C2)$ can be defined as

$$(C2): \sum_{e_k \in E_w} p_{e_k} \leq d_{G\backslash E_w(v_i,v_j)}(v_i, v_j) \quad \forall i,j \ \ 0 \leq i < j \leq n \tag{13}$$

In $(C2)$, (v_i, v_j) is a vertex pair from V_w. $G\backslash E_w(v_i, v_j)$ is the graph removing the edges of $E_w(v_i, v_j)$ from G, and $d_{G\backslash E_w(v_i,v_j)}(v_i, v_j)$ is the cost of the shortest path from v_i to v_j in this graph. Note that there may exist no path from v_i to v_j after removing $E_w(v_i, v_j)$, in which case the constraint corresponding to the vertex pair (v_i, v_j) is not included in $(C2)$.

In Fig. 2, the constraints of $(C2)$ are

$$\begin{cases} p_{e_3} \leq 3 \\ p_{e_1} + p_{e_2} \leq 5 \\ p_{e_1} + p_{e_2} + p_{e_3} \leq 8 \end{cases} \tag{14}$$

For example, (v_0, v_2) is a vertex pair and E_w is the edge set $\{e_1, e_2\}$. Due to $d_{G\backslash\{e_1,e_2\}}(v_0, v_2) = 5$, we have $p_{e_1} + p_{e_2} \leq 5$ according to (13). We can see that the core constraint number is much smaller than $(C1)$. Given that $|V_w| = n + 1$, the number of vertex

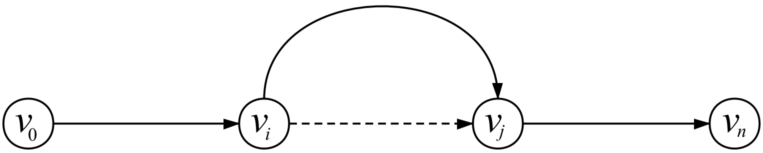

Fig. 3 The path $P_{G\setminus E_w(v_i,v_j)}(v_0, v_n)$, signed by solid arrows

pair is at most $\frac{n(n+1)}{2}$, which means we only need to run the shortest path algorithm $\frac{n(n+1)}{2}$ times. Therefore, the computational complexity can be reduced greatly.

Theorem 2 *The two constraint sets* $(C1)$ *and* $(C2)$ *describe the same core.*

This theorem means that $(C2)$ is equivalent to $(C1)$. Then, we will prove Theorem 2 from two aspects, i.e. necessity and sufficiency. Necessity is to prove $(C1) \Rightarrow (C2)$ and sufficiency is to prove $(C2) \Rightarrow (C1)$.

3.1 Necessity: $(C1) \Rightarrow (C2)$

To begin with, we provide two lemmas for the shortest path.

Lemma 1 *Given the source vertex and the target vertex, the cost of the shortest path is not longer than other walks in the graph.*

Lemma 2 *A subpath of a shortest path is itself a shortest path.*

In Fig. 3, (v_i, v_j) is an arbitrary vertex pair from $V_w(v_0, v_n)$. The dotted arrow represents $P_w(v_i, v_j)$. The curved arrow represents the shortest path $P_{G\setminus E_w(v_i,v_j)}(v_i, v_j)$, whose cost is $d_{G\setminus E_w(v_i,v_j)}(v_i, v_j)$. Then we can find a path which is $v_0 \to v_i \to v_j \to v_n$[1] signed by solid arrows in Fig. 9. This path exists after removing $E_w(v_i, v_j)$ and its cost is $d_{G\setminus E_w(v_i,v_j)}(v_i, v_j) + (d_G(v_0, v_n) - \sum_{e_k \in E_w(v_i,v_j)} c_{e_k})$. According to Lemma 1, we have

$$d_{G\setminus E_w(v_i,v_j)}(v_0, v_n) \le d_{G\setminus E_w(v_i,v_j)}(v_i, v_j) + d_G(v_0, v_n) - \sum_{e_k \in E_w(v_i,v_j)} c_{e_k} \tag{15}$$

In $(C1)$, let $x = E_w(v_i, v_j)$ and we can obtain

$$\sum_{e_k \in E_w(v_i,v_j)} p_{e_k} \le d_{G\setminus E_w(v_i,v_j)}(v_0, v_n) - \left(d_G(v_0, v_n) - \sum_{e_k \in E_w(v_i,v_j)} c_{e_k} \right) \tag{16}$$

Substituting (15) into (16), we get

[1] We use $v_i \to v_j$ to represent the path from v_i to v_j in the preceding text.

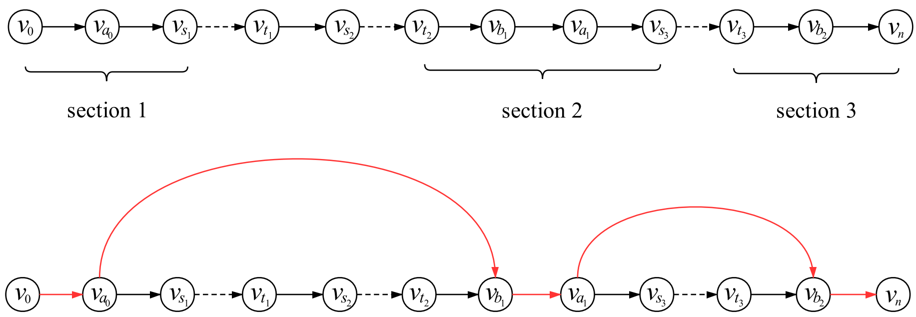

Fig. 4 A situation including the shortest path in graph G and $G\backslash x$. Top: The straight path from v_0 to v_n represents the path $P_G(v_0, v_n)$, and the dotted arrows represent the removed subset x; Bottom: The path signed with the red arrows represents the path $P_{G\backslash x}(v_0, v_n)$ (Color figure online)

$$\sum_{e_k \in E_w(v_i,v_j)} p_{e_k} \le d_{G\backslash E_w(v_i,v_j)}(v_i, v_j) \tag{17}$$

We can see that inequality (17) is just a constraint in $(C2)$. Vertex pair (v_i, v_j) is arbitrary so that we prove $(C1) \Rightarrow (C2)$.

Next, we prove the sufficiency in the following three parts. Sections 3.2 and 3.3 describe the prerequisite theorems and Sect. 3.4 describes the final mathematical proof using path division.

3.2 Sufficiency: preparation theorems

Lemma 3 *Assuming that $G_1 \subset G$, if a path is the shortest path in G and it also exists in G_1, then this path is also the shortest path in G_1.*

Given Lemma 3, we start by discussing the arbitrary subset x in $(C1)$. Without loss of generality, x is a union of some edge sets, which can be written as $x = E_w(v_{s_1}, v_{t_1}) \cup E_w(v_{s_2}, v_{t_2}) \cup \cdots \cup E_w(v_{s_m}, v_{t_m})$, where $0 \le s_1 < t_1 < s_2 < \cdots < t_m \le n$. After removing x, the shortest path is divided into several sections. These sections represent the subpaths of the shortest path, which are $P_w(v_0, v_{s_1}), P_w(v_{t_1}, v_{s_2}), \ldots, P_w(v_{t_m}, v_n)$ [2]. Then we have that $v_0 \in P_w(v_0, v_{s_1})$ and $v_n \in P_w(v_{t_m}, v_n)$. Besides, these sections mutually disjoint with each other because they are divided by x. According to Lemmas 2 and 3, we can see that each subpath of these sections is also the shortest path in graph $G\backslash x$. We mainly consider these sections in the proof.

A situation is described as Fig. 4, where $x = E_w(v_{s_1}, v_{t_1}) \cup E_w(v_{s_2}, v_{t_2}) \cup E_w(v_{s_3}, v_{t_3})$. The shortest path is divided into four disjoint sections that are $P_w(v_0, v_{s_1})$, $P_w(v_{t_1}, v_{s_2})$, $P_w(v_{t_2}, v_{s_3})$ and $P_w(v_{t_3}, v_n)$. Each section represents a subpath of $P_w(v_0, v_n)$.

$P_{G\backslash x}(v_0, v_n)$ is the shortest path in $G\backslash x$, abbreviated as $P_{G\backslash x}$ for convenience. In Fig. 4, $P_{G\backslash x}$ is signed by the red arrows. According to the definition, $P_{G\backslash x}$ can't include any edge in x, but it may include some edges in some sections. Then, we provide two theorems for this sort of section.

[2] Notice that $P_w(v_0, v_{s_1})$ may be a subpath from v_0 to v_0 when $s_1 = 0$, and so is $P_w(v_{t_m}, v_n)$.

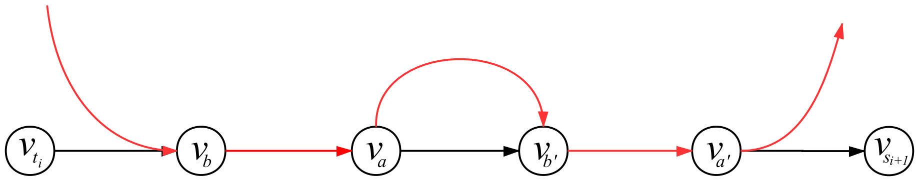

Fig. 5 A counter-example for Proposition 1

Theorem 3 *If a section $P_w(v_{t_i}, v_{s_{i+1}})$ $(0 < i < m)$ has common edges with $P_{G\backslash x}$, there exists at least one common vertex v_a satisfying*

1. *in the path $P_{G\backslash x}$, the edge ending at v_a belongs to $P_w(v_{t_i}, v_{s_{i+1}})$;*
2. *in the path $P_{G\backslash x}$, the edge starting at v_a doesn't belong to $P_w(v_{t_i}, v_{s_{i+1}})$.*

Proof Assume that there exists no vertex meeting the requirements. Denote by v_i the ending vertex of one common edge. In $P_{G\backslash x}$, the edge starting at v_i must belong to $P_w(v_{t_i}, v_{s_{i+1}})$, otherwise, v_i is just the vertex we are looking for. Denote by v_{i+1} the next vertex and $v_{i+1} \in P_w(v_{t_i}, v_{s_{i+1}})$. Similarly, the edge starting at v_{i+1} also belongs to $P_w(v_{t_i}, v_{s_{i+1}})$, so the next vertex v_{i+2} also belongs to $P_w(v_{t_i}, v_{s_{i+1}})$. Keep deriving and we will find that all the vertices after v_i belong to $P_w(v_{t_i}, v_{s_{i+1}})$. Due to $i < m$, this conflicts with the factor that $v_n \notin P_w(v_{t_i}, v_{s_{i+1}})$. As a consequence, Theorem 3 is established. □

For the sake of convenience, we denote by $\mathcal{A}$ the two conditions above. In Fig. 4, v_{a_1} is a vertex satisfying $\mathcal{A}$. Similarly, we have the following theorem.

Theorem 4 *If a section $P_w(v_{t_i}, v_{s_{i+1}})$ $(0 < i < m)$ has common edges with $P_{G\backslash x}$, there exists at least one common vertex v_b satisfying*

1. *in the path $P_{G\backslash x}$, the edge ending at v_b doesn't belong to $P_w(v_{t_i}, v_{s_{i+1}})$;*
2. *in the path $P_{G\backslash x}$, the edge starting at v_b belongs to $P_w(v_{t_i}, v_{s_{i+1}})$.*

The proof is similar to Theorem 3. Also, denote by $\mathcal{B}$ the two conditions above. In Fig. 4, the vertex v_{b_1} satisfies $\mathcal{B}$.

3.3 Sufficiency: properties for the vertices satisfying $\mathcal{A}$ and $\mathcal{B}$

According to Theorems 3 and 4, if a section $P_w(v_{t_i}, v_{s_{i+1}})$ has common edges with $P_{G\backslash x}$, then we can find a vertex v_a satisfying $\mathcal{A}$ and a vertex v_b satisfying $\mathcal{B}$. Denote by $v_i \rightarrow v_j$ the subpath from v_i to v_j in $P_{G\backslash x}$. For the vertex v_a, we have the following proposition.

Proposition 1 *In $P_{G\backslash x}$, the subpath $v_a \rightarrow v_n$ has no common edges with $P_w(v_a, v_{s_{i+1}})$.*

Proof Assuming that Proposition 1 is wrong. The situation can be described as Fig. 5. $P_{G\backslash x}$ is signed by the red arrows and the edge $(v_{b'}, v_{a'})$ is a common edge between the subpath $v_a \rightarrow v_n$ and $P_w(v_a, v_{s_{i+1}})$. We know that the shortest path from v_a to $v_{a'}$ is the straight path $P_w(v_a, v_{a'})$, which belongs to $P_w(v_a, v_{s_{i+1}})$. However, due to that the edge starting at

v_a doesn't belong to $P_w(v_{t_i}, v_{s_{i+1}})$, the subpath $v_a \to v_{a'}$ is different from the shortest path $P_w(v_a, v_{a'})$. If these two paths have different costs, this will produce a contradiction according to Lemma 2. Therefore, these two paths must have the same costs. In this case, we just replace the path $v_a \to v_{a'}$ with $P_w(v_a, v_{a'})$ to update the path $P_{G\backslash x}$. With the eventual update of $P_{G\backslash x}$, the proposition will become true and we use this path in our discussion. □

In Fig. 4, as to the vertex v_{a_1}, Proposition 1 means the subpath $v_{a_1} \to v_n$ has no common edges with $P_w(v_{a_1}, v_{s_3})$. On the basis of symmetry, we have a similar proposition for vertex v_b.

Proposition 2 *In $P_{G\backslash x}$, the subpath $v_0 \to v_b$ has no common edges with $P_w(v_{t_i}, v_b)$.*

The proof is similar to Proposition 1. In Fig. 4, as to the vertex v_{b_1}, Proposition 2 means the subpath $v_0 \to v_{b_1}$ has no common edges with $P_w(v_{t_2}, v_{b_1})$.

3.4 Sufficiency: path division

Based on the conclusion in Sects. 3.2 and 3.3, we could consider the traveling process of path $P_{G\backslash x}$.

The path $P_{G\backslash x}$ starts at $v_0, v_0 \in P_w(v_{t_0}, v_{s_1})$, then it may pass some edges in the section $P_w(v_{t_0}, v_{s_1})$ and leave this section after passing a vertex satisfying $\mathcal{A}$, denoted by v_{a_0}. Otherwise, $P_{G\backslash x}$ may not pass any edge in $P_w(v_{t_0}, v_{s_1})$ and leave this section directly, in which case we let $a_0 = 0$. After leaving this section, path $P_{G\backslash x}$ will arrive at another section $P_w(v_{t_i}, v_{s_{i+1}})$ $(i > 0)$, which is the first section having common edges with $P_{G\backslash x}$ after passing v_{a_0}. If $i \neq m$, then $P_{G\backslash x}$ arrives at a vertex v_{b_1} satisfying $\mathcal{B}$. After that, it will pass some edges in $P_w(v_{t_i}, v_{s_{i+1}})$ and leave $P_w(v_{t_i}, v_{s_{i+1}})$ at a vertex v_{a_1} that satisfies $\mathcal{A}$. After leaving the previous section, $P_{G\backslash x}$ will arrive at the next section and repeat the same process until this path reaches the last section $P_w(v_{t_m}, v_n)$. Finally, $P_{G\backslash x}$ will pass a vertex v_{b_k} satisfying $\mathcal{B}$ and reach the target vertex v_n through the subpath $P_w(v_{b_k}, v_n)$, which is the shortest path. Or $P_{G\backslash x}$ may reach the target vertex v_n directly, in which case we let $b_k = n$.

Note that $b_0 = 0$ and $a_k = n$. By the traveling process, $P_{G\backslash x}$ can be divided into

$$v_{b_0} \to v_{a_0} \to v_{b_1} \to v_{a_1} \dots v_{b_{k-1}} \to v_{a_{k-1}} \to v_{b_k} \to v_{a_k}$$

For example, in Fig. 4, the path division is $v_0 \to v_{a_0} \to v_{b_1} \to v_{a_1} \to v_{b_2} \to v_n$. The passed sections are $P_w(v_0, v_{s_1}), P_w(v_{t_2}, v_{s_3}), P_w(v_{t_3}, v_n)$. These sections are signed by section 1, 2 and 3 in Fig. 4.

According to Lemmas 2 and 3, these subpaths are the shortest in graph $G\backslash x$. We consider a part of these subpaths as

$$v_{b_0} \to v_{a_0}, v_{b_1} \to v_{a_1}, \dots, v_{b_k} \to v_{a_k}$$

The two endpoints of these subpaths are in the same section, so the shortest path between them in $G\backslash x$ is the same as that in G. Let $U = \bigcup_{i=0}^{k} E_w(v_{b_i}, v_{a_i})$, and the total cost of these subpaths above will be $\sum_{e_i \in U} c_{e_i}$.

Then we consider the rest subpaths as

$$v_{a_0} \to v_{b_1}, v_{a_1} \to v_{b_2}, \dots, v_{a_{k-1}} \to v_{b_k}$$

We can see that each subpath $v_{a_i} \to v_{b_{i+1}}$ is a subpath of the path $v_0 \to v_{b_{i+1}}$ and $v_{a_i} \to v_n$. Due to Propositions 1, 2 and that $v_{b_{i+1}}$ belongs to the first section which has common edges

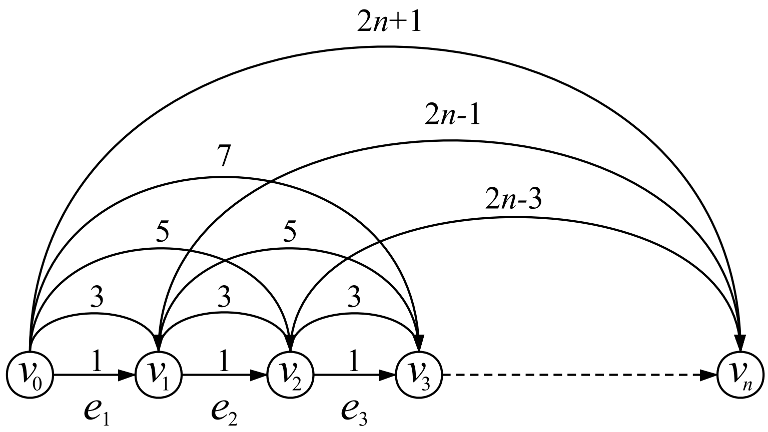

Fig. 6 A specific graph, where the number on each edge is its cost

after passing v_{a_i}, we can draw a conclusion that subpath $v_{a_i} \rightarrow v_{b_{i+1}}$ has no common edges with $E_w(v_{a_i}, v_{b_{i+1}})$. Therefore, $v_{a_i} \rightarrow v_{b_{i+1}}$ still exists in the graph $G \backslash E_w(v_{a_i}, v_{b_{i+1}})$.

Denote by D_i the cost of subpath $v_{a_i} \rightarrow v_{b_{i+1}}$. $(v_{a_i}, v_{b_{i+1}})$ is a vertex pair in constraint set ($C2$). According to Lemma 1, the constraint corresponding to this vertex pair in ($C2$) becomes

$$\sum_{e_l \in E_w(v_{a_i}, v_{b_{i+1}})} p_{e_l} \leq d_{G \backslash E_w(v_{a_i}, v_{b_{i+1}})}(v_{a_i}, v_{b_{i+1}}) \leq D_i, \forall i \in [0, k-1] \tag{18}$$

Combining k inequalities in (18), we have

$$\sum_{e_i \in E_w \backslash U} p_{e_i} \leq \sum_{i=0}^{k-1} D_i \tag{19}$$

According to the previous definition, we have $\sum_{i=0}^{k-1} D_i = d_{G \backslash x}(v_0, v_n) - \sum_{e_i \in U} c_{e_i}$. As a result, (19) can be written as

$$\sum_{e_i \in E_w \backslash U} (p_{e_i} - c_{e_i}) \leq d_{G \backslash x}(v_0, v_n) - d_G(v_0, v_n) \tag{20}$$

We can see that $x \subset E_w \backslash U$ because U and x both are the subsets of E_w and $U \cap x = \emptyset$. According to the IR constraint $p_{e_i} - c_{e_i} \geq 0$, we can obtain the following inequality.

$$\sum_{e_i \in x} (p_{e_i} - c_{e_i}) \leq \sum_{e_i \in E_w \backslash U} (p_{e_i} - c_{e_i}) \tag{21}$$

Substituting (20) into (21), we obtain

$$\sum_{e_i \in x} (p_{e_i} - c_{e_i}) \leq d_{G \backslash x}(v_0, v_n) - d_G(v_0, v_n) \tag{22}$$

Inequality (22) is the same as constraint (10) in ($C1$), which holds for each subset x. Thus, the sufficiency is proved and Theorem 2 is established.

3.5 Explanation for the optimality

On the basis of conclusion above, we know that the constraint set ($C2$) can produce the core correctly. However, ($C2$) may also have some redundant constraints. To verify the optimality of ($C2$), we construct a worst case as shown in Fig. 6.

Definition 6 *(Redundant constraint)* A constraint is redundant if the feasible domain does not change after removing it from the constraint set.

In Fig. 6, the length of the shortest path is n. Firstly, we consider a simple situation including vertices v_0, v_1, v_2, v_3 and the connected directed edges among them. If the auctioneer wants to buy a path from v_0 to v_3, then the winner set will be $E_w = \{e_1, e_2, e_3\}$ and the payment set will be $P = \{p_{e_1}, p_{e_2}, p_{e_3}\}$. The constraints in ($C2$) will be

$$\begin{cases} p_{e_1} \leq 3,\ p_{e_2} \leq 3,\ p_{e_3} \leq 3 \\ p_{e_1} + p_{e_2} \leq 5,\ p_{e_2} + p_{e_3} \leq 5 \\ p_{e_1} + p_{e_2} + p_{e_3} \leq 7 \end{cases} \tag{23}$$

To produce the core, we also need IR constraints $p_{e_1} \geq 1, p_{e_2} \geq 1, p_{e_3} \geq 1$. These constraints form a complete core constraint set. According to Theorem 5, it is easy to verify that none of these constraints is redundant.

Theorem 5 *For a constraint, if we can find a payment result beyond the core, which becomes feasible after removing this constraint, then this constraint is not redundant.*

For arbitrary n, where n is the number of winners, we can construct such an example that all the constraints of ($C2$) are not redundant in this example. Therefore, we have the following theorem.

Theorem 6 *To produce the core correctly, the number of constraints is at least* $\frac{n^2}{2} + \frac{3}{2}n$, *where n is the number of winners.*

This theorem is proved in [6] and we omit it in this paper. In addition, the total size of constraint set ($C2$) and IR constraints is exactly $\frac{n^2}{2} + \frac{3}{2}n$, which means this constraint set is the optimal core constraint set for path auction.

4 A new format of constraint set ($C3$)

Though the number of constraints couldn't be reduced, the format of constraint can be organized to accelerate the computation. For the purpose of computation, we put forward a new format of constraint set ($C3$) as

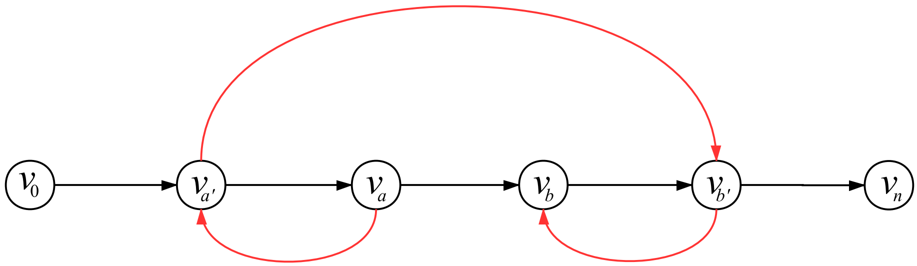

Fig. 7 An example of vertex pair (v_a, v_b) and the path $P_{G\backslash E_w(v_a,v_b)}(v_a, v_b)$ is signed by red arrows (Color figure online)

$$(C3):\sum_{e_k\in E_w(v_i,v_j)} p_{e_k} \le d_{G\backslash E_w}(v_i, v_j) \quad \forall i,j\ 0 \le i < j \le n \tag{24}$$

where $G\backslash E_w$ is the graph that removes the edges of E_w in G. For any vertex pair (v_i, v_j), if $d_{G\backslash E_w}(v_i, v_j)$ doesn't exist, then the corresponding constraint won't be included in $(C3)$. In Fig. 2, for example, the constraints of $(C3)$ are the same as $(C2)$.

According to $(C3)$, to get the core is to find the edge weights of the shortest path such that every subpath of the shortest path should be no more than the second shortest path after removing the shortest path. In graph theory language, we could summary that the core is the change range of weights of the shortest path such that the shortest path remains the shortest between two given vertices.

In this format, the number of constraints is no more than $(C2)$ because when $d_{G\backslash E_w(v_i,v_j)}(v_i, v_j)$ doesn't exist, $d_{G\backslash E_w}(v_i, v_j)$ doesn't exist either. $(C3)$ is just like another format of $(C2)$ and it can reduce some redundant constraints of $(C2)$. Moreover, $(C3)$ is convenient because the shortest path is only computed in one graph $G\backslash E_w$, instead of different graphs in $(C2)$. Then we can accelerate the computation by using some fast algorithms such as single-source shortest path algorithm. Above all, $(C3)$ is better than $(C2)$ in general case. Next, we prove the correctness of $(C3)$.

Theorem 7 *The two constraint sets* $(C2)$ *and* $(C3)$ *describe the same core.*

Proof Similarly, we prove this theorem from two aspects and we prove $(C2) \Rightarrow (C3)$ at first. □

Lemma 4 *Assuming that there exists at least a path from* v_i *to* v_j *in both* G *and* G_1*, if* $G \supset G_1$*, then* $d_G(v_i, v_j) \le d_{G_1}(v_i, v_j)$.

For any vertex pair (v_a, v_b) where $0 \le a < b \le n$, we have that $G\backslash E_w \subset G\backslash E_w(v_a, v_b)$. According to Lemma 4, we have the following inequality.

$$\sum_{e_k\in E_w(v_a,v_b)} p_{e_k} \le d_{G\backslash E_w(v_a,v_b)}(v_a, v_b) \le d_{G\backslash E_w}(v_a, v_b) \tag{25}$$

Therefore, $(C2) \Rightarrow (C3)$ is established because (25) holds for arbitrary vertex pair (v_a, v_b). Afterwards, we prove that $(C3) \Rightarrow (C2)$.

As in Fig. 7, (v_a, v_b) is an arbitrary vertex pair. The constraint corresponding to (v_a, v_b) in $(C2)$ is

$$\sum_{e_k \in E_w(v_a,v_b)} p_{e_k} \leq d_{G \backslash E_w(v_a,v_b)}(v_a, v_b) \tag{26}$$

$P_{G\backslash E_w(v_a,v_b)}(v_a, v_b)$ is the shortest path from v_a to v_b in $G\backslash E_w(v_a, v_b)$ and $P_{G\backslash E_w}(v_a, v_b)$ is the shortest path in $G\backslash E_w$. If the two paths have the same cost, it is obvious that $(C3) \Rightarrow (C2)$ is established.

In the other case that the two paths' costs are different, $P_{G\backslash E_w(v_a,v_b)}(v_a, v_b)$ is shorter than $P_{G\backslash E_w}(v_a, v_b)$ because that there are more edges in graph $G\backslash E_w(v_a, v_b)$. These extra edges are from the edge set $E_w(v_0, v_a)$ and $E_w(v_b, v_n)$, so the path $P_{G\backslash E_w(v_a,v_b)}(v_a, v_b)$ must pass some edges in $E_w(v_0, v_a) \cup E_w(v_b, v_n)$, otherwise the two paths' costs will be the same. Therefore, it must pass some vertices of $V_w(v_0, v_a) \cup V_w(v_b, v_n)$. We use $v_{a'}$ to denote the last passed vertex in $V_w(v_0, v_a)$. Due to $v_a \in V_w(v_0, v_a)$, the vertex $v_{a'}$ must exist. After passing $v_{a'}$, this path will pass another vertex $v_{b'}$ which is the first passed vertex in $V_w(v_b, v_n)$. The vertex $v_{b'}$ must exist because $v_b \in V_w(v_b, v_n)$. Then, we obtain a subpath from $v_{a'}$ to $v_{b'}$, which is the shortest path in $G\backslash E_w(v_a, v_b)$. This subpath is also the shortest in $G\backslash E_w$ according to Lemma 3. Thus, in the constraint set $(C3)$, the constraint corresponding to the vertex pair $(v_{a'}, v_{b'})$ is

$$\sum_{e_k \in E_w(v_{a'},v_{b'})} p_{e_k} \leq d_{G \backslash E_w}(v_{a'}, v_{b'}) \tag{27}$$

We can see that $E_w(v_a, v_b) \subset E_w(v_{a'}, v_{b'})$, then the constraint can be written as

$$\sum_{e_i \in E_w(v_a,v_b)} p_{e_i} \leq d_{G \backslash E_w}(v_{a'}, v_{b'}) \tag{28}$$

$d_{G\backslash E_w}(v_{a'}, v_{b'})$ is the cost of a subpath of $P_{G\backslash E_w(v_a,v_b)}(v_a, v_b)$, so we have

$$d_{G \backslash E_w}(v_{a'}, v_{b'}) \leq d_{G \backslash E_w(v_a,v_b)}(v_a, v_b) \tag{29}$$

Finally, by substituting (29) into (28), we obtain

$$\sum_{e_k \in E_w(v_a,v_b)} p_{e_k} \leq d_{G \backslash E_w(v_a,v_b)}(v_a, v_b) \tag{30}$$

The constraint (30) is just the constraint of vertex pair (v_a, v_b) in $(C2)$. (v_a, v_b) is arbitrary, so $(C3) \Rightarrow (C2)$ is proved and Theorem 7 is established. □

5 Core pricing algorithms

Previous sections are the theoretical discussion of the constraint sets in core-selecting path mechanism. As we know, the core gives us a feasible region constrained by a set of inequalities, then, how to choose the final result in the feasible region is the next question to answer.

To minimize incentives of misreporting, a minimum-revenue core-selecting rule is put forward and commonly used as a pricing rule for spectrum auction [9]. This rule chooses the bidder-Pareto-optimal core outcome as the final result, which could maximize the profit of bidders and don't block the core constraints. Next, we provide several pricing algorithms to get the bidder-Pareto-optimal core outcome and discuss them in the rest of this paper.

Definition 7 *(Bidder-Pareto-optimal core outcome)* A core outcome is bidder-Pareto-optimal if there is no other core outcome weakly preferred by every bidder in the winner coalition.

Theorem 8 *An outcome is bidder-Pareto-optimal if it owns the maximum total payment in the core.*

Proof When the total payment is the maximum in the core, there exists no outcome that could improve one's profit without hurting others' profits in the winner coalition. Therefore, the outcome with a maximum payment is bidder-Pareto-optimal in the core. □

In graph theory language, this outcome is aimed to find the largest total weight such that the original shortest path could remain the shortest. As we can see, the problem of computing the maximum total payment is just an optimization problem subject to the core constraints as follows.

$$\begin{aligned} &\max \sum_{e_i \in E_w} p_{e_i} \\ &\textit{subject to} : \textit{core constraints} \end{aligned} \tag{31}$$

Since all the constraints are linear inequalities, the straightforward method is to solve a linear programming problem after obtaining all the constraints. Following this idea, we design three direct algorithms that are Linear Programming Path Auction algorithm based on the constraint set ($C1$), ($C2$) and ($C3$) respectively, abbreviated as LPPA-C1, LPPA-C2 and LPPA-C3 algorithm.

5.1 LPPA-C1 algorithm

Review the constraint set ($C1$)

$$\sum_{e_i \in x} p_{e_i} \le d_{G\backslash x}(v_0, v_n) - d_G(v_0, v_n) + \sum_{e_i \in x} c_{e_i} \tag{32}$$

Let β_x^1 be $d_{G\backslash x}(v_0, v_n) - d_G(v_0, v_n) + \sum_{e_i \in x} c_{e_i}$ for each x. Note that $\beta_x^1 = \infty$ if $d_{G\backslash x}(v_0, v_n)$ doesn't exist. All β_x^1 form a vector $\boldsymbol{\beta}^1$, then we have

$$\boldsymbol{A}^1 \boldsymbol{p}^T \le {\boldsymbol{\beta}^1}^T \tag{33}$$

The formula (33) represents ($C1$). $\boldsymbol{A}^1$ is a $(2^n - 1) * n$ matrix, where n is the edge number of the shortest path. $\boldsymbol{A}_{ij}^1$ equals 1 only when the i-th set x includes the j-th edge e_j, or $\boldsymbol{A}_{ij}^1$ equals 0. $\boldsymbol{p}$ is the vector of payment profile. Then the maximum payment can be computed by solving linear programming LP^1.

$$\begin{aligned} LP^1 &: \alpha = max\, \boldsymbol{p} \times \boldsymbol{1} \\ subject\ to &: \boldsymbol{A}^1 \boldsymbol{p}^T \leq {\boldsymbol{\beta}^1}^T \\ &\boldsymbol{p} \geq \boldsymbol{c} \end{aligned} \tag{34}$$

$\boldsymbol{c}$ is the cost vector and α is the maximum payment. LP^1 has n decision variables and $2^n - 1 + n$ constraints. The pseudocode of LPPA-C1 algorithm is described as Algorithm 1.

Algorithm 1 Linear Programming Path Auction algorithm based on $(C1)$

Input: Directed graph $G = (V, E, C)$ (in G each edge e_i has a nonnegative weight $c_{e_i}, c_{e_i} \in C$); source vertex v_0; target vertex v_n; the winner set E_w;

Output: Maximum $\sum_{e_i \in E_w} p_{e_i}$

1: C1-SET $\leftarrow \{p_{e_i} \geq c_{e_i} | e_i \in E_w\}$
2: // C1-SET is the constraint set for LPPA-C1 algorithm.
3: **for** $x \in E_w$ **do**
4: Compute the value of A^1_x and β^1_x
5: Add the corresponding constraint into C1-SET
6: **end for**
7: Solve LP^1 and get a payment $P \leftarrow \{p_{e_1}, p_{e_2}, \ldots, p_{e_n}\}$
8: **return** $\sum_{e_i \in E_w} p_{e_i}$

5.2 LPPA-C2 algorithm

Review the constraint set $(C2)$

$$\sum_{e_k \in E_w(v_a, v_b)} p_{e_k} \leq d_{G \backslash E_w(v_a, v_b)}(v_a, v_b) \tag{35}$$

(v_a, v_b) is a vertex pair of V_w and $0 \leq a < b \leq n$. Similarly, let $\beta^2_{(a,b)}$ be $d_{G \backslash E_w(v_a, v_b)}(v_a, v_b)$ for each pair (v_a, v_b). Note that $\beta^2_{(a,b)} = \infty$ if $d_{G \backslash E_w(v_a, v_b)}(v_a, v_b)$ doesn't exist. All $\beta^2_{(a,b)}$ form a vector $\boldsymbol{\beta}^2$, then we have

$$\boldsymbol{A}^2 \boldsymbol{p}^T \leq {\boldsymbol{\beta}^2}^T \tag{36}$$

The formula (36) represents $(C2)$. $\boldsymbol{A}^2$ is a $\frac{n(n+1)}{2} \times n$ matrix, where n is the edge number of the shortest path. $\boldsymbol{A}^2_{ij}$ equals 1 only when the j-th edge e_j is in the i-th set $E_w(v_a, v_b)$ or $\boldsymbol{A}^2_{ij}$ equals 0. Then the maximum payment can be computed by solving linear programming LP^2

$$\begin{aligned} LP^2 &: \alpha = max\, \boldsymbol{p} \times \boldsymbol{1} \\ subject\ to &: \boldsymbol{A}^2 \boldsymbol{p}^T \leq {\boldsymbol{\beta}^2}^T \\ &\boldsymbol{p} \geq \boldsymbol{c} \end{aligned} \tag{37}$$

LP^2 has n decision variables and $\frac{n(n+1)}{2} + n$ constraints. Compared with LPPA-C1 algorithm, the constraint number greatly decreases. The procedure is similar so we omit it.

5.3 LPPA-C3 algorithm

Review the constraint set ($C3$)

$$\sum_{e_k \in E_w(v_a, v_b)} p_{e_k} \leq d_{G \backslash E_w}(v_a, v_b) \tag{38}$$

(v_a, v_b) is a vertex pair of V_w and $0 \leq a < b \leq n$. Similarly, let $\beta^3_{(a,b)}$ be $d_{G \backslash E_w}(v_a, v_b)$ for each pair (v_a, v_b). Note that $\beta^3_{(a,b)} = \infty$ if $d_{G \backslash E_w}(v_a, v_b)$ doesn't exist. All $\beta^3_{(a,b)}$ form a vector $\boldsymbol{\beta}^3$, then we have

$$\boldsymbol{A}^3 \boldsymbol{p}^T \leq {\boldsymbol{\beta}^3}^T \tag{39}$$

The formula (39) represents ($C3$). $\boldsymbol{A}^3$ is a $\frac{n(n+1)}{2} \times n$ matrix, where n is the edge number of the shortest path. $\boldsymbol{A}^3_{ij}$ equals 1 only when the j-th edge e_j is in the i-th set $E_w(v_a, v_b)$, or $\boldsymbol{A}^3_{ij}$ equals 0. Then we can calculate the maximum payment by solving linear programming LP^3 .

$$\begin{aligned} LP^3 &: \alpha = max\, \boldsymbol{p} \times \boldsymbol{1} \\ subject\ to &: \boldsymbol{A}^3 \boldsymbol{p}^T \leq {\boldsymbol{\beta}^3}^T \\ & \quad \boldsymbol{p} \geq \boldsymbol{c} \end{aligned} \tag{40}$$

LP^3 has n decision variables and $\frac{n(n+1)}{2} + n$ constraints, which is the same as LPPA-C2 algorithm. We notice that single-source shortest path algorithm could be used to accelerate the calculation in LPPA-C3 algorithm.

Algorithm 2 Linear Programming Path Auction algorithm based on ($C3$)

Input: Directed graph $G = (V, E, C)$ (in G each edge e_i has a nonnegative weight $c_{e_i}, c_{e_i} \in C$); source vertex v_0; target vertex v_n; the winner set E_w;
Output: Maximum $\sum_{e_i \in E_w} p_i$
C3-SET $\leftarrow \{p_{e_i} \geq c_{e_i} | e_i \in E_w\}$
Remove the edges in E_w, get the graph $G \backslash E_w$
for $i \in [1, n-1]$ **do**
  Make v_i the source vertex, run single-source shortest path algorithm in graph $G \backslash E_w$.
  // This algorithm is used to compute the costs of the shortest paths from v_i to all reachable vertices.
  for $j \in [i+1, n]$ **do**
    C3-SET $\leftarrow$ C3-SET $\cup \{\sum_{e_i \in E_w(v_i, v_j)} p_{e_i} \leq d_{E_w}(v_i, v_j)\}$
    // Note that $d_{E_w}(v_i, v_j) \leftarrow +\infty$ when $d_{E_w}(v_i, v_j)$ isn't obtained.
  end for
end for
Solve LP^3 and get a payment $P \leftarrow \{p_{e_1}, p_{e_2}, \ldots, p_{e_n}\}$
return $\sum_{e_i \in E_w} p_{e_i}$

In LPPA-C1 algorithm and LPPA-C2 algorithm, the shortest path is calculated in different graphs $G \backslash x$ and $G \backslash E_w(v_a, v_b)$. The graph will be different when x or (v_a, v_b) is different. Therefore, LPPA-C1 algorithm needs to run the shortest path algorithm $2^n - 1$ times while LPPA-C2 needs to run $\frac{n(n+1)}{2}$ times. By contrast, LPPA-C3 algorithm is convenient because that all the shortest paths are computed in the same graph $G \backslash E_w$. As a result, we can use single-source or multi-source shortest path algorithm, which is faster according to our experiments.

In this paper we use the single-source shortest path algorithm to compute all the costs.[3] LPPA-C3 algorithm just needs to run this shortest path algorithm $(n-1)$ times. The pseudocode is described as Algorithm 2.

5.4 CCG-VCG algorithm

To get the bidder-Pareto-optimal core outcome in a combinatorial auction, Core Constraint Generation (CCG) algorithm was proposed by [12], which is expected to deal with a moderate number of constraints. CCG algorithm uses the method of constraint generation that considers only the most valuable constraints. According to [12], we design a specific CCG algorithm called CCG-VCG algorithm for path auction.

Definition 8 *(Most blocking path)* For an outcome O including E_w and P, replace the bids in E_w with the payments in P and denote by $E_{w'}$ the new winner set. If the total cost of edges in $E_{w'}$ is equal to the total value in P, then O has no blocking paths. Otherwise, $E_{w'}$ is the most blocking path for the outcome O.

Algorithm 3 CCG-VCG algorithm for Path Auction

Input: Directed graph $G=(V,E,C)$ (in G each edge e_i has a nonnegative weight $c_{e_i}, c_{e_i} \in C$); source vertex v_0; target vertex v_n; the winner set E_w;
Output: Maximum $\sum_{e_i \in E_w} p_{e_i}$

$t \leftarrow 0$
Compute VCG payment of each winner e_i, denoted as $p^t_{e_i}$
CCG-SET $\leftarrow \{p_{e_i} \le p^t_{e_i}, p_{e_i} \ge c_{e_i} | e_i \in E_w\}$
// CCG-SET is the constraint set for CCG algorithm.
$\forall e_i \in E_w, c_{e_i} \leftarrow p^t_{e_i}$
Compute the new winner set $E_{w'}$.
while $\sum_{e_i \in E_w} p^t_{e_i} \ne \sum_{e_i \in E_{w'}} c_{e_i}$ **do**
    $t \leftarrow t+1$
    $z \leftarrow E_w \cap E_{w'}$
    CCG-SET $\leftarrow$ CCG-SET $\cup \{\sum_{e_i \in E_w \setminus z} p_{e_i} \le \sum_{e_i \in E_{w'} \setminus z} c_{e_i}\}$
    Solve the problem LP and get a payment $P^t = \{p^t_{e_1}, p^t_{e_2}, \ldots, p^t_{e_n}\}$.
    $\forall e_i \in E_w,\ c_{e_i} \leftarrow p^t_{e_i}$
    Compute the new winner set $E_{w'}$.
end while
return $\sum_{e_i \in E_w} p^t_{e_i}$

In CCG-VCG algorithm, we initialize the payments using the VCG price for each winner. Denote by CCG-SET the constraint set in this algorithm and we initialize CCG-SET as follows.

[3] The algorithm is used by the package nexworkx 2.1, we didn't use the multi-source shortest path algorithm because that actually networkx 2.1 achieves the multi-source algorithm by calling single-source algorithm many times.

$$\text{CCG-SET}\begin{cases} p_{e_i} \le p_{e_i}^{VCG}, & 1 \le i \le n \\ p_{e_i} \ge c_{e_i} \quad , & 1 \le i \le n \end{cases} \tag{41}$$

The iterations of the algorithm are indexed by t and $p_{e_i}^t$ is the payment for e_i in the t-th iteration. We compute the winner set $E_{w'}$ after this initialization. If the total cost of $E_{w'}$ is not equal to the total payment of $p_{e_i}^t$, then we have a most blocking path. Let $z = E_w \cap E_{w'}$ and we obtain a constraint related to $E_{w'}$ as

$$\sum_{e_i \in E_w \setminus z} p_{e_i} \le \sum_{e_i \in E_{w'} \setminus z} c_{e_i} \tag{42}$$

Add the constraint (42) into CCG-SET and solve the linear programming problem LP.

$$\begin{aligned} \text{LP}:\ & max \quad \boldsymbol{P} \times \mathbf{1} \\ subject\ to:\ & \quad \text{CCG-SET} \end{aligned} \tag{43}$$

Afterwards, we could get a payment $P^t = \{p_{e_1}^t, p_{e_2}^t, \dots, p_{e_n}^t\}$ as the result of LP. P^t is the payment set in the t-th iteration. Then we change the cost of winner edges from $p_{e_i}^{t-1}$ to $p_{e_i}^t$ and compute the new winner set $E_{w'}$. If $E_{w'}$ is a most blocking path then we start a new iteration, otherwise we will get the final result P^t, which is a bidder-Pareto-optimal core outcome.

The pseudocode of CCG-VCG algorithm is shown above as Algorithm 3 and the proof for its correctness is provided in the appendix.

6 Bellman–Ford path auction algorithm

As we can see, the core is a polytope with polynomial constraints and the optimization objective of maximum total payment is a hyperplane. This objective should be tangent to the polytope when we get the maximum payment. Thus, there must be some tight constraints that are necessary to consider. Under such motivation, we propose a novel algorithm called Bellman–Ford Path Auction (BFPA) algorithm. BFPA algorithm can get the maximum core payment by only running Bellman–Ford algorithm once. Bellman–Ford

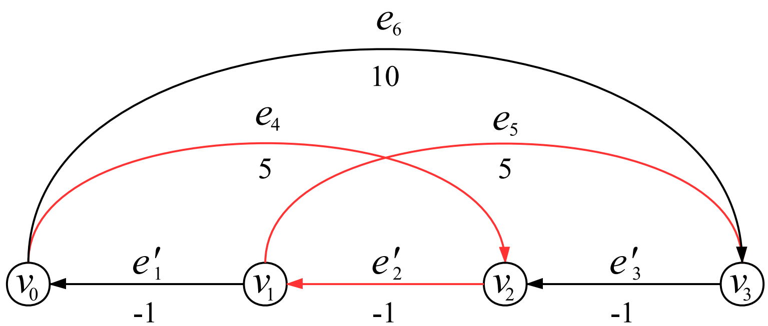

Fig. 8 An example for BFPA algorithm. There are 6 bidders $e'_1, e'_2, e'_3, e_4, e_5, e_6$ with cost $-1, -1, -1, 5, 5, 10$, where e'_i is the reverse edge of the winner e_i

algorithm[4] is a single-source shortest path algorithm that can deal with the edges which have negative weights.

Algorithm 4 Bellman-Ford Path Auction algorithm

Input: Directed graph $G = (V, E, C)$ (in G each edge e_i has a nonnegative weight $c_{e_i}, c_{e_i} \in C$); source vertex v_0; target vertex v_n; the winner set E_w;
Output: Maximum $\sum_{e_i \in E_w} p_{e_i}$

Delete all the edges in E_w
for $e_i \in E_w$ **do**
    Add a reverse edge of e_i with cost of $-c_{e_i}$
end for
// Get the graph G'.
Make v_0 the source vertex, run Bellman-Ford algorithm in graph G'.
// This algorithm is used to compute the cost of the shortest paths from v_i to all reachable vertices.
for $i \in [1, n]$ **do**
    $p_{e_i} \leftarrow d_{G'}(v_0, v_i) - d_{G'}(v_0, v_{i-1})$
end for
return $\sum_{e_i \in E_w} p_{e_i}$

BFPA algorithm is described as Algorithm 4. In BFPA algorithm, we generate a graph G' by converting each edge in E_w to a reverse edge with a negative cost. Compute the shortest path from v_0 to other vertices in V_w and we can obtain a solution of payment as

$$p_{e_i} = d_{G'}(v_0, v_i) - d_{G'}(v_0, v_{i-1}) \quad \forall i \in [1, n] \tag{44}$$

$d_{G'}(v_0, v_i)$ is the cost of the shortest path from v_0 to v_i in G'. Note that $d_{G'}(v_0, v_0) = 0$, and the total payment of this solution is

$$\begin{aligned} \sum_{i=1}^{n} p_{e_i} &= \sum_{i=1}^{n} (d_{G'}(v_0, v_i) - d_{G'}(v_0, v_{i-1})) \\ &= d_{G'}(v_0, v_n) - d_{G'}(v_0, v_0) \\ &= d_{G'}(v_0, v_n) \end{aligned} \tag{45}$$

For example, in Fig. 8, the original winner set E_w is $\{e_1, e_2, e_3\}$ and each winner's cost is 1. In BFPA algorithm, we delete the edges in E_w and add their reverse edges as e'_1, e'_2, e'_3 to get the graph G'. We can see that the shortest path from v_0 to v_1 is $P_{G'}(v_0, v_1) = v_0 e_4 v_2 e'_2 v_1$, whose cost is $d_{G'}(v_0, v_1) = 4$. Similarly, we have $d_{G'}(v_0, v_2) = 5$ and $d_{G'}(v_0, v_3) = 9$. Then we could get the payment solution of BFPA algorithm as

$$\begin{cases} p_{e_1} = 4 - 0 = 4 \\ p_{e_2} = 5 - 4 = 1 \\ p_{e_3} = 9 - 5 = 4 \end{cases} \tag{46}$$

[4] In the experiment, the algorithm we use is actually SPFA algorithm, it is an improved version of Bellman–Ford algorithm.

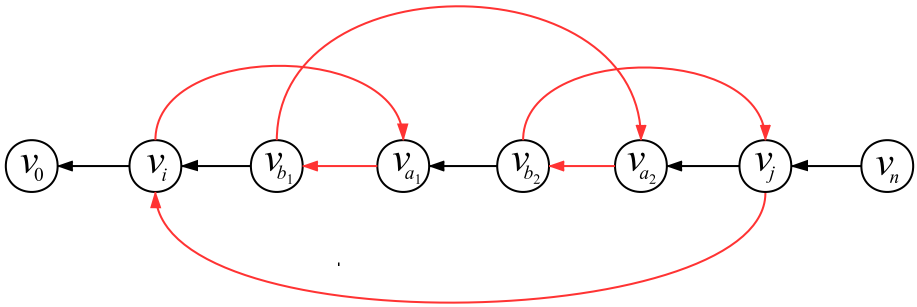

Fig. 9 Negative circuit $\mathcal{C}$, signed by red arrows (Color figure online)

Next, we completely prove that the payment solution of BFPA algorithm is a correct maximum core payment in Sects. 6.1–6.3.

6.1 Existence of the shortest path in graph G'

In the beginning, we assume that there is no cut edge for the connectivity from v_0 to v_n, which could not obviously draw the conclusion that there exists a path from v_0 to v_n in G'. Therefore, we need to prove the following theorem at first.

Theorem 9 *There exists at least one path from v_0 to v_n in graph G'.*

The proof is provided in the appendix. Once there are edges with negative costs in the graph, there is doubt whether a negative closed walk exists. If a negative closed walk exists, there may not exist the shortest walk because it can pass the negative closed walk endlessly so that the cost can be infinitely negative. Bellman–Ford algorithm couldn't work in this situation.

Lemma 5 *Any closed walk can be expressed as a union of several circuits, where circuit is a closed walk with no repeated vertices.*

According to Lemma 5, the following Theorem 10 is sufficient to guarantee the non-existence of negative closed walk in G'.

Theorem 10 *There exists no negative circuit in G'.*

Proof Assuming that there exists one negative circuit in G', denoted by $\mathcal{C}$. $\mathcal{C}$ must include some negative edges in G' so that it must pass some vertices in V_w. Among all the passed vertices in V_w, denote by v_i the leftmost vertex and v_j the rightmost vertex, like Fig. 9. This circuit has two subpaths that are $v_i \rightarrow v_j$ and $v_j \rightarrow v_i$. We only consider the subpath $v_i \rightarrow v_j$.

This subpath can only pass negative edges between v_i and v_j. Without loss of generality, assume that the first passed negative edge is (v_{a_1}, v_{a_1-1}). Then this subpath may pass some vertices in $V_w(v_i, v_{a_1})$ and denote by v_{b_1} the last passed vertex. v_{b_1} must exist because

$v_{a_1-1} \in V_w(v_i, v_{a_1})$ and it is passed after v_{a_1}. After that, this path may also pass some negative edges between v_{a_1} and v_j. Denote by (v_{a_2}, v_{a_2-1}) the first passed negative edge and v_{b_2} the last passed vertex in $V_w(v_{a_1}, v_{a_2})$. Repeat this process until this path finally ends at vertex v_j. Note that $a_0 = b_0 = i, a_x = b_x = j$, and we can get a path division as

$$v_{b_0} \rightarrow v_{a_1} \rightarrow v_{b_1} \rightarrow v_{a_2} \rightarrow v_{b_2} \dots v_{b_{x-1}} \rightarrow v_{a_x}$$

For example, in Fig. 9, the path division is $v_i \rightarrow v_{a_1} \rightarrow v_{b_1} \rightarrow v_{a_2} \rightarrow v_{b_2} \rightarrow v_j$. In this path division, v_{a_2} is the starting vertex of the first passed negative edge between v_{a_1} and v_j. v_{b_2} is the last passed vertex in $V_w(v_{a_1}, v_{a_2})$.

We consider one part of these subpaths as

$$v_{b_0} \rightarrow v_{a_1}, v_{b_1} \rightarrow v_{a_2}, \dots, v_{b_{x-1}} \rightarrow v_{a_x}$$

These subpaths can be unified as $v_{b_i} \rightarrow v_{a_{i+1}}, i \in [0, x-1]$. As the definition above, the last passed vertex in $V_w(v_{a_0}, v_{a_1}), V_w(v_{a_1}, v_{a_2}), \dots, V_w(v_{a_{i-1}}, v_{a_i})$ is $v_{b_1}, v_{b_2}, \dots, v_{b_i}$. Therefore, v_{b_i} is not only the last passed vertex in $V_w(v_{a_{i-1}}, v_{a_i})$ but also the last passed vertex in $V_w(v_{a_0}, v_{a_i})$. Besides, $v_{a_{i+1}}$ is the starting vertex of the first passed negative edge between v_{a_i} and v_j. As a consequence, the subpath $v_{b_i} \rightarrow v_{a_{i+1}}$ doesn't include any negative edge, so these subpaths also exist in graph G. Denote by $d_{G'}(v_{b_i} \rightarrow v_{a_{i+1}})$ the cost of the subpath $v_{b_i} \rightarrow v_{a_{i+1}}$ and we can obtain the following inequalities.

$$\begin{cases} \sum_{e_k \in E_w(v_{b_0}, e_{v_{a_1}})} c_{e_k} \le d_{G'}(v_{b_0} \rightarrow v_{a_1}) \\ \sum_{e_k \in E_w(v_{b_1}, v_{a_2})} c_{e_k} \le d_{G'}(v_{b_1} \rightarrow v_{a_2}) \\ \dots \\ \sum_{e_k \in E_w(v_{b_{x-1}}, v_{a_x})} c_{e_k} \le d_{G'}(v_{b_{x-1}} \rightarrow v_{a_x}) \end{cases} \tag{47}$$

Combine these x inequalities and we obtain

$$\sum_{l=0}^{x-1} \sum_{e_k \in E_w(v_{b_l}, v_{a_{l+1}})} c_{e_k} \le \sum_{l=0}^{x-1} d_{G'}(v_{b_l} \rightarrow v_{a_{l+1}}) \tag{48}$$

Due to that $a_l \ge b_l, \forall l \in [0, x]$, we have the following inequality.

$$\sum_{l=0}^{x-1} \sum_{e_k \in E_w(v_{b_l}, v_{a_{l+1}})} c_{e_k} \ge \sum_{l=0}^{x-1} \sum_{e_k \in E_w(v_{b_l}, v_{b_{l+1}})} c_{e_k} = \sum_{e_k \in E_w(v_i, v_j)} c_{e_k} \tag{49}$$

Substituting (49) into (48), we obtain

$$\sum_{e_k \in E_w(v_i, v_j)} c_{e_k} \le \sum_{l=0}^{x-1} d_{G'}(v_{b_l} \rightarrow v_{a_{l+1}}) \tag{50}$$

In this inequality, the right side is the sum of some positive subpaths' costs in the circuit $\mathcal{C}$ and the left side is the sum of absolute value of all the negative edges' costs between v_i and v_j. This circuit passes at most all the negative edges between v_i and v_j. Denote by $d_{G'}(\mathcal{C})$ the cost of circuit $\mathcal{C}$ and we have

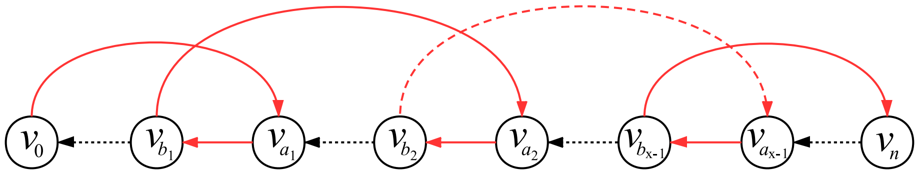

Fig. 10 The path $P_{G'}$, signed by red arrows (Color figure online)

$$d_{G'}(C) \geq -\sum_{e_k \in E_w(v_i,v_j)} c_{e_k} + \sum_{l=0}^{x-1} d_{G'}(v_{b_l} \to v_{a_{l+1}}) \geq 0 \tag{51}$$

Inequality (51) means that the cost of this circuit can't be negative, which produces a contradiction, so Theorem 10 is established. □

Lemma 6 *If there isn't any negative circuit in the graph G', then $\forall s, t \in G$, the cost of the shortest walk from s to t is equal to that of the shortest path.*

Since there isn't a negative circuit in graph G', according to Lemma 6, we only need to consider the paths in graph G'. We also have Theorem 11.

Theorem 11 *The shortest path from v_n to v_0 in G' is the reverse path of original shortest path $P_w(v_0, v_n)$.*

Proof This reverse path includes all the negative edges in G', so it is the path with the smallest cost. There is no other path shorter than it. Therefore, it is the shortest path. □

We denote by $P_{G'}(v_n, v_0)$ this reverse path. According to Lemma 2, each subpath of $P_{G'}(v_n, v_0)$ is also the shortest path in G'.

6.2 Upper bound of the core outcome

In this part, we only provide a proof for Theorem 12.

Theorem 12 *$d_{G'}(v_0, v_n)$ is an upper bound for total payment in the core.*

Proof $d_{G'}(v_0, v_n)$ is the cost of the shortest path from v_0 to v_n in G'. We denote this path by $P_{G'}(v_0, v_n)$, abbreviated as $P_{G'}$. Just like the analysis in Sect. 6.1, let $i = 0, j = n$ and we can get a division of $P_{G'}$ as

$$v_0 \to v_{a_1} \to v_{b_1} \to v_{a_2} \to v_{b_2}, \ldots, v_{b_{x-1}} \to v_n$$

Figure 10 is a general situation of $P_{G'}$. v_{a_i} is the starting vertex of the first passed negative edge between $v_{a_{i-1}}$ and v_n while v_{b_i} is the last passed vertex in $V_w(v_{a_{i-1}}, v_{a_i})$. Then we consider a part of subpaths as

$$v_{b_0} \to v_{a_1}, v_{b_1} \to v_{a_2}, \dots, v_{b_{x-1}} \to v_{a_x}$$

Note that $a_0 = b_0 = 0$ and $a_x = b_x = n$. According to the previous analysis, there isn't any negative edge in these subpaths, so these subpaths still exist in graph $G \backslash E_w$. Denote by $d_{G'}(v_{b_i} \to v_{a_{i+1}})$ the cost of the subpath $v_{b_i} \to v_{a_{i+1}}$, $i \in [0, x-1]$. Based on the constraint set $(C3)$ and Lemma 1, we could get a constraint for the core payment as

$$\begin{aligned} \sum_{e_k \in E_w(v_{b_i}, v_{a_{i+1}})} p_{e_k} &\le d_{G \backslash E_w}(v_{b_i}, v_{a_{i+1}}) \\ &\le d_{G'}(v_{b_i} \to v_{a_{i+1}}) \end{aligned} \tag{52}$$

where $i \in [0, x-1]$. Due to that $b_i < b_{i+1} < a_{i+1}$, combine these x constraints above and we can get

$$\begin{aligned} \sum_{i=0}^{x-1} \sum_{e_k \in E_w(v_{b_i}, v_{a_{i+1}})} p_{e_k} &= \sum_{i=0}^{x-1} \sum_{e_k \in E_w(v_{b_i}, v_{b_{i+1}})} p_{e_k} + \sum_{i=0}^{x-1} \sum_{e_k \in E_w(v_{b_{i+1}}, v_{a_{i+1}})} p_{e_k} \\ &= \sum_{e_k \in E_w} p_{e_k} + \sum_{i=1}^{x-1} \sum_{e_k \in E_w(v_{b_i}, v_{a_i})} p_{e_k} \\ &\le \sum_{i=0}^{x-1} d_{G'}(v_{b_i} \to v_{a_{i+1}}) \end{aligned} \tag{53}$$

Inequality (53) can be reorganized as

$$\sum_{e_k \in E_w} p_{e_k} \le \sum_{i=0}^{x-1} d_{G'}(v_{b_i} \to v_{a_{i+1}}) - \sum_{i=1}^{x-1} \sum_{e_k \in E_w(v_{b_i}, v_{a_i})} p_{e_k} \tag{54}$$

Then, we consider the rest subpaths as

$$v_{a_1} \to v_{b_1}, v_{a_2} \to v_{b_2}, \dots, v_{a_{x-1}} \to v_{b_{x-1}}$$

These subpaths are also the shortest because $P_{G'}$ is the shortest path. According to Theorem 11 and Lemma 2, the shortest path from v_{a_i} to v_{b_i} should be the reverse path of $P_w(v_{b_i}, v_{a_i})$, and its cost is $-\sum_{e_k \in E_w(v_{b_i}, v_{a_i})} c_{e_k}$. Denote by $d_{G'}(v_{a_i} \to v_{b_i})$ the cost of subpath $v_{a_i} \to v_{b_i}$, where $i \in [1, x-1]$, and we have the following equation.

$$d_{G'}(v_{a_i} \to v_{b_i}) = -\sum_{e_k \in E_w(v_{b_i}, v_{a_i})} c_{e_k} \tag{55}$$

According to individual rationality, we have

$$\sum_{e_k \in E_w(v_{b_i}, v_{a_i})} p_{e_k} \ge \sum_{e_k \in E_w(v_{b_i}, v_{a_i})} c_{e_k} = -d_{G'}(v_{a_i} \to v_{b_i}) \tag{56}$$

Combining all the constraints in (56) for $i = 1, 2, \dots, x-1$, we obtain

$$\sum_{i=1}^{x-1} \sum_{e_k \in E_w(v_{b_i}, v_{a_i})} p_{e_k} \ge -\sum_{i=1}^{x-1} d_{G'}(v_{a_i} \to v_{b_i}) \tag{57}$$

Finally, substituting (57) into (54), we can get

$$\begin{aligned}\sum_{e_k \in E_w(v_0,v_n)} p_{e_k} &\leq \sum_{i=0}^{x-1} d_{G'}(v_{b_i} \rightarrow v_{a_{i+1}}) + \sum_{i=1}^{x-1} d_{G'}(v_{a_i} \rightarrow v_{b_i}) \\ &= d_{G'}(v_{b_0} \rightarrow v_{a_1}) + d_{G'}(v_{a_1} \rightarrow v_{b_1}) + \cdots + d_{G'}(v_{b_{x-1}} \rightarrow v_{a_x}) \\ &= d_{G'}(v_0, v_n)\end{aligned} \tag{58}$$

Formula (58) is derived through the constraints of $(C3)$ and individual rationality. This formula means that the sum of core payment is no more than $d_{G'}(v_0, v_n)$. Thus, we can draw a conclusion that $d_{G'}(v_0, v_n)$ is an upper bound of total payment in the core. □

6.3 The proof for constraint satisfaction

In Sect. 6.2, we have proved that BFPA algorithm achieves a total payment of $d_{G'}(v_0, v_n)$, which is an upper bound in the core. But this is not enough to verify that it is a correct core outcome. BFPA algorithm gives a payment for each winner, which is $p_{e_i} = d_{G'}(v_0, v_i) - d_{G'}(v_0, v_{i-1})$. Next, we will prove that this payment solution satisfies all the constraints in core-selecting path mechanism so that it is a correct core outcome.

Theorem 13 *The payment solution of BFPA algorithm satisfies all the IR constraints, i.e.*

$$p_{e_i} \geq c_{e_i}, \forall i \in [1, n]$$

Proof Assuming that there exists a IR constraint which is not satisfied, then this constraint will become

$$p_{e_i} < c_{e_i},\ i \in [1, n] \tag{59}$$

According to BFPA algorithm, we have

$$p_{e_i} = d_{G'}(v_0, v_i) - d_{G'}(v_0, v_{i-1}) < c_{e_i} \tag{60}$$

Reorganize (60) and we obtain

$$d_{G'}(v_0, v_i) + (-c_{e_i}) < d_{G'}(v_0, v_{i-1}) \tag{61}$$

In the inequality (61), $d_{G'}(v_0, v_{i-1})$ is the cost of the shortest path from v_0 to v_{i-1} in G'. In the left side, $d_{G'}(v_0, v_i)$ is the cost of the shortest path from v_0 to v_i in G', denoted by $v_0 \rightarrow v_i$, and $(-c_{e_i})$ is the cost of a negative edge (v_i, v_{i-1}) in G'. Thus, $d_{G'}(v_0, v_i) + (-c_{e_i})$ is the cost of a path from v_0 to v_{i-1}, which consists of two subpaths that are $v_0 \rightarrow v_i$ and (v_i, v_{i-1}). According to (61), its cost is less than $d_{G'}(v_0, v_{i-1})$. This is contradictory to that $d_{G'}(v_0, v_{i-1})$ is the cost of the shortest path. Therefore, the payment solution of BFPA algorithm satisfies all the IR constraints. □

Theorem 14 *The payment solution of BFPA algorithm satisfies all the constraints in* $(C3)$, *i.e.*

$$\sum_{e_k \in E_w(v_i,v_j)} p_{e_k} \leq d_{G \backslash E_w}(v_i, v_j) \quad \forall i,j \;\; 0 \leq i < j \leq n$$

Proof Assuming that there exists a constraint in $(C3)$ which is not satisfied, then this constraint will become

$$\sum_{e_k \in E_w(v_i,v_j)} p_{e_k} > d_{G \backslash E_w}(v_i, v_j) \quad 0 \leq i < j \leq n \tag{62}$$

According to BFPA algorithm, we have

$$\begin{aligned} \sum_{e_k \in E_w(v_i,v_j)} p_{e_k} &= \sum_{k=i+1}^{j} (d_{G'}(v_0, v_k) - d_{G'}(v_0, v_{k-1})) \\ &= d_{G'}(v_0, v_j) - d_{G'}(v_0, v_i) \end{aligned} \tag{63}$$

Substituting (63) into (62), we obtain the following inequality.

$$d_{G'}(v_0, v_j) > d_{G'}(v_0, v_i) + d_{G \backslash E_w}(v_i, v_j) \tag{64}$$

In (64), $d_{G'}(v_0, v_j)$ is the cost of the shortest path from v_0 to v_j in G'. In the right side, $d_{G'}(v_0, v_i)$ is the cost of the shortest path from v_0 to v_i in G', denoted by $v_0 \to v_i$. $d_{G \backslash E_w}(v_i, v_j)$ is the cost of the shortest path from v_i to v_j in $G \backslash E_w$ and this path also exists in G' because $G \backslash E_w \subset G'$, denoted by $v_i \to v_j$. Thus, $d_{G'}(v_0, v_i) + d_{G \backslash E_w}(v_i, v_j)$ is just the cost of a walk from v_0 to v_j in G'. This walk consists of two paths that are $v_0 \to v_i$ and $v_i \to v_j$. According to (64), its cost is less than $d_{G'}(v_0, v_j)$. Similarly, this is contradictory to that $d_{G'}(v_0, v_j)$ is the cost of the shortest path (also the shortest walk). Therefore, the payment solution of BFPA algorithm satisfies all the constraints in $(C3)$. □

Above all, we can draw a conclusion that the payment solution of BFPA algorithm is a correct core outcome and its total payment is the maximum in the core. Therefore, this payment solution is just a bidder-Pareto-optimal core outcome we are looking for.

7 Experiment

7.1 Experiment dataset

We choose five network datasets in SNAP [25] to construct the graphs in our experiments. These networks are described as follows.

- *Facebook network*
  The dataset consists of friend lists from Facebook. The data was collected from survey participants using Facebook app.
- *Wikipedia voting network*
  The network contains voting data for Wikipedia administrator elections. Vertices in the network represent Wikipedia users and a directed edge from vertex i to vertex j represents that user i votes on user j.
- *P2p-Gnutella04 network*

Table 1 Network statistics

Networks	Vertices	Edges	d_{max}	90-per-centile d_{max}
Facebook	4039	88,234	8	4.7
Wiki-Vote	7115	103,689	7	3.8
P2p-Gnutella04	10,876	39,994	9	5.4
Soc-Epinions1	75,879	508,837	14	5
Twitter	81,306	1,768,149	7	4.5

Table 2 Average total payment under reported cost distribution

Algorithm	Facebook	Wiki-Vote	P2p04	Soc-Ep1	Twitter
Shortest path	2803.11 (± 424.28)	1117.08 (± 250.86)	5882.30 (± 473.79)	1817.86 (± 302.22)	1321.71 (± 138.68)
VCG	8499.39 (± 1873.64)	2951.42 (± 624.32)	19033.63 (± 2549.37)	6293.88 (± 1179.04)	3548.40 (± 711.23)
Core pricing algorithm					
CCG-VCG	5756.58 (± 1096.39)	2585.18 (± 607.04)	11215.42 (± 1253.28)	5415.19 (± 992.79)	2926.54 (± 650.81)
LPPA-C2	5756.58 (± 1096.39)	2585.18 (± 607.04)	11215.42 (± 1253.28)	5415.19 (± 992.79)	2926.54 (± 650.81)
LPPA-C3	5756.58 (± 1096.39)	2585.18 (± 607.04)	11215.42 (± 1253.28)	5415.19 (± 992.79)	2926.54 (± 650.81)
BFPA	5756.58 (± 1096.39)	2585.18 (± 607.04)	11215.42 (± 1253.28)	5415.19 (± 992.79)	2926.54 (± 650.81)

The dataset describes the Gnutella peer-to-peer file sharing network from August 4 2002. Vertices represent hosts in the Gnutella network topology and edges represent connections between the Gnutella hosts.

- *Soc-Epinions1 network*
  This is a who-trust-whom online social network of a general consumer review site Epinions.com. The vertices represent the members of the site and the edges represent their trust relationships.
- *Twitter network*
  This dataset consists of friend lists from Twitter. The data was crawled from public sources.

The detailed network statistics are given in Table 1.

In these networks, the true information of cost is hard to get. In this paper, we use reported cost data from a micro-blog advertising platform weiboyi,[5] where micro-bloggers

[5] http://www.weiboyi.com/.

Table 3 Worst case time complexity

Algorithm	Worst case time complexity
Shortest path algorithm	$O(\|E\| + \|V\| \log \|V\|)$
VCG	$O(n(\|E\| + \|V\| \log \|V\|))$
Core pricing algorithm	
CCG-VCG	$O((n + m)(\|E\| + \|V\| \log \|V\|))$
LPPA-C1	$O(2^n(\|E\| + \|V\| \log \|V\|))$
LPPA-C2	$O(n^2(\|E\| + \|V\| \log \|V\|))$
LPPA-C3	$O(n(\|E\| + \|V\| \log \|V\|))$
BFPA	$O(\|V\|\|E\|)$

are asked to report their cost to make recommendations to friends in their social networks. We assign the edges randomly with costs in this dataset.

In each network, we generate 200 problem instances where the source vertex v_0 and target vertex v_n are selected uniformly at random from all vertices. The experiments were run on a high-performance server with 20×2.2 GHZ Intel Xeon Opteron cores and 230 GB of RAM. Besides, we describe the graph with networkx 2.1 and solve the linear programming with SciPy 0.19.1 on a runtime environment of Python 3.6.8. The payment result is shown in Table 2. Note that the bracketed content is the confidence interval of 95%.

In Table 2, we can see that the total payment computed by all the core pricing algorithms is always the same in all networks. This result confirms the correctness of our algorithms. Besides, the maximum core total payment is smaller than VCG total payment, which indicates that core-selecting path mechanism can reduce the overpayment.

7.2 Computational efficiency

In this part, we mainly demonstrate the computational efficiency of our algorithms. We use two shortest path algorithms in this paper, which are Dijkstra algorithm and Bellman–Ford algorithm. Their time complexities are $O(|E| + |V| \log |V|)$ and $O(|V||E|)$ respectively. Then, we provide the summary of time complexity for our algorithms in Table 3, where n is the number of winners and m is the number of iterations in CCG-VCG algorithm. Notice that there is a process of linear programming in core pricing algorithms except BFPA algorithm, whose time complexity is $O(n^2) \sim O(n^3)$. We omit this time cost because it is very small compared with the shortest path algorithms in our experiments.

As shown in Table 3, LPPA-C1 algorithm is significantly slower than the other algorithms so that we don't compare its performance with other methods in our experiments. According to Table 3, the time complexity of LPPA-C3 algorithm is $O(n(|E| + |V| \log |V|))$, which is strictly better than CCG-VCG algorithm and LPPA-C2 algorithm. In addition, we can see that the time complexity of BFPA algorithm is $O(|V||E|)$, which is independent of n.

Afterwards, we compare the average runtime in our experiments of five real-world datasets. The result is shown in Table 4. LPPA-C2 algorithm and CCG-VCG algorithm are the state-of-the-art algorithms achieved in the previous work [6]. Compared with these two algorithms, our new algorithms (LPPA-C3 and BFPA) have better performance in all datasets. As the fastest algorithm, BFPA algorithm is about $1 \sim 5$ times faster than LPPA-C2 algorithm and $1 \sim 4$ times faster than CCG-VCG algorithm. Besides, it is faster than computation of VCG payment.

Table 4 Average runtime performance (in s)

Algorithm	Facebook	Wiki-Vote	P2p04	Soc-Ep1	Twitter
Shortest path	0.019	0.040	0.027	0.406	1.331
	(±0.003)	(±0.005)	(±0.003)	(±0.030)	(±0.105)
VCG	0.172	0.266	0.469	4.946	24.528
	(±0.029)	(±0.021)	(±0.049)	(±0.312)	(±1.880)
Core pricing algorithm					
CCG-VCG	0.366	0.511	0.842	9.163	41.280
	(±0.059)	(±0.039)	(±0.083)	(±0.645)	(±3.511)
LPPA-C2	0.540	0.518	1.916	8.258	44.043
	(±0.103)	(±0.053)	(±0.270)	(±1.011)	(±6.305)
LPPA-C3	0.191	0.308	0.595	4.498	21.601
	(±0.023)	(±0.014)	(±0.038)	(±0.214)	(±1.084)
BFPA	**0.116**	**0.197**	**0.285**	**2.864**	**10.616**
	(±0.012)	(±0.011)	(±0.009)	(±0.091)	(±0.326)

Bold values indicate the fastest runtime among all the core pricing algorithms in each of the columns

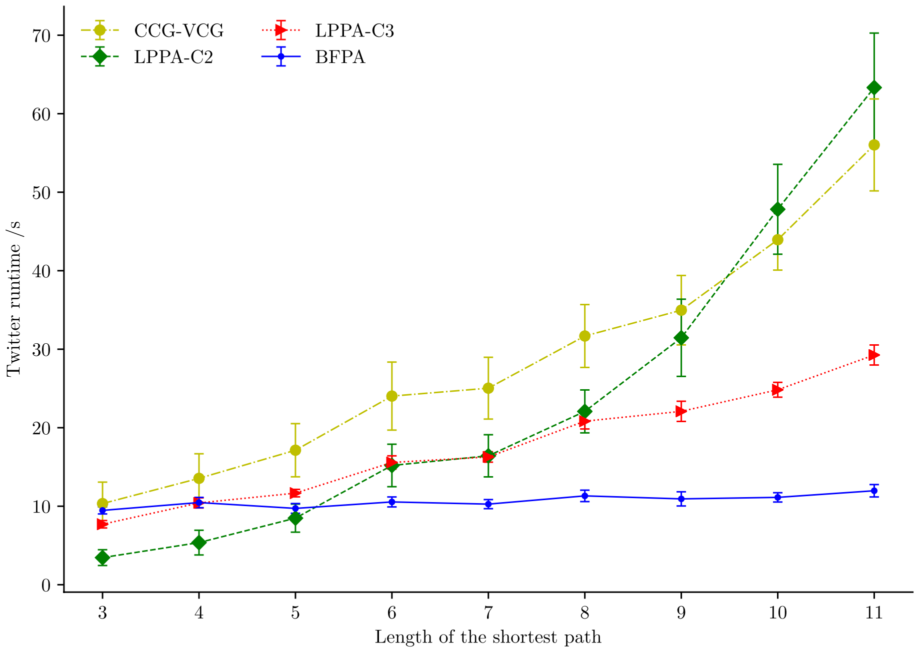

Fig. 11 The relationship between the runtime and the length of the shortest path in Twitter network, the error bar represents the confidence interval of 95%

In order to explore the relationship between the runtime and the length of the shortest path, we carried on another experiment in the largest Twitter network. In this experiment, we randomly selected 50 vertex pairs based on different lengths of the shortest path to run the algorithms. The result is shown in Fig. 11.

On the whole, our two new algorithms perform better than the other core pricing algorithms. BFPA algorithm may be a little slower when the length is short and it is the fastest when the length is more than 5. Besides, we can see that the runtime increases with the increasing length of path. The increasing length has a remarkable impact on LPPA-C2 algorithm and CCG-VCG algorithm, while the impact is smaller in LPPA-C3 algorithm. As to BFPA algorithm, such impact is little and its runtime remains horizontal basically. This is because the primary time cost of BFPA algorithm is the Bellman–Ford algorithm, whose process is only related to the starting vertex. These experiments also verify our analysis of time complexity. Notice that LPPA-C2 and LPPA-C3 could compute the complete core while CCG-VCG and BFPA only get a final core outcome.

8 Conclusion

In this paper, we focus on the core pricing algorithms in path auction. Although the winner determination problem is easy to solve in path auction, there are also exponential constraints to consider. Firstly, we reduce the constraint number from $O(2^n)$ to $O(n^2)$ theoretically, and we prove that the number of constraints is at least $\frac{n^2}{2} + \frac{3}{2}n$ to produce the core correctly. Then, we give a novel format of constraint set ($C3$) which is convenient to obtain the core constraint by using single-source shortest path algorithm. Besides, in order to get the bidder-Pareto-optimal core outcome, we achieve five algorithms including LPPA-C1, LPPA-C2, LPPA-C3, CCG-VCG and BFPA. Last but not least, we test these algorithms in real-world datasets and our two new algorithms (LPPA-C3 and BFPA) significantly surpass the other algorithms in terms of runtime.

In our opinions, this paper reveals some structure properties of path auction and core polytope. Therefore, one of our future works is to improve the computation of core-selecting mechanism in other combinatorial auctions, whether to accelerate CCG algorithm or put forward new algorithms by using this heuristic approach. The other is to consider the payment in core-selecting path auction. In reality, the most commonly used approach is the core outcome which is nearest to the VCG payment [9], but some current researches show that this approach may not be the best payment rule [5]. As a result, it is valuable to work on it.

Acknowledgements This paper is supported by the National Key Research and Development Program of China (Grant No. 2018YFB1403400), the National Natural Science Foundation of China (Grant No. 61876080), the Collaborative Innovation Center of Novel Software Technology and Industrialization at Nanjing University.

Appendix A: Summary of main notation

$G = (V, E)$	A directed graph that consists of a edge set E and a vertex set V

v_0	The starting vertex
v_n	The ending vertex
$e_i = (v_{i-1}, v_i)$	The edge that starts in the vertex v_{i-1} and ends in v_i, also represents a bidder owning edge e_i
c_{e_i}	The cost of the edge e_i
π_{e_i}	The utility of the bidder e_i
Π	Social welfare of the auction
0	The auctioneer
N	The total set of players, including the bidders and auctioneer
$W(L)$	The social welfare of the subset L of N
P	The payment set of the auction
p_{e_i}	The payment to the bidder e_i
core	The total set of core outcomes
$W_G(v_i, v_j)$	A walk from v_i to v_j in graph G
$P_G(v_i, v_j)$	The shortest path from v_i to v_j in graph G
$V_G(v_i, v_j)$	The vertex set of $P_G(v_i, v_j)$
$E_G(v_i, v_j)$	The edge set of $P_G(v_i, v_j)$
$P_w(v_0, v_n)$	The path that is selected as the winner path in the auction
$E_w(v_0, v_n)$ *or* E_w	The edge set of $P_w(v_0, v_n)$
$V_w(v_0, v_n)$ *or* V_w	The vertex set of $P_w(v_0, v_n)$
$P_w(v_i, v_j)$	A subpath of $P_w(v_0, v_n)$ that is from v_i to v_j
$E_w(v_i, v_j)$	The edge set of $P_w(v_i, v_j)$
$V_w(v_i, v_j)$	The vertex set of $P_w(v_i, v_j)$
$d_G(v_i, v_j)$	The cost of the shortest path from v_i to v_j in graph G

Appendix B: Proof of the correctness of CCG-VCG algorithm for path auction

Theorem 15 *The outcome of CCG-VCG algorithm is a bidder-Pareto-optimal core outcome.*

Proof Consider the constraint added into CCG-SET

$$\sum_{e_i \in E_w \setminus z} p_{e_i} \leq \sum_{e_i \in E_{w'} \setminus z} c_{e_i} \tag{65}$$

$E_{w'}$ is the new winner set in a new graph where we change the cost of each edge in E_w from $p_{e_i}^{t-1}$ to $p_{e_i}^t$ in G. Denote by G_1 this graph and $P_{w'}$ the path corresponding to the edge set $E_{w'}$. $P_{w'}$ is the shortest path in G_1. We first prove that the constraint (65) is a standard constraint of $(C1)$.

In G_1, we remove the edges in $E_w \setminus z$ and change the costs of the edges in z from $p_{e_i}^t$ to c_{e_i}. Denote by G_2 this graph. $P_{w'}$ also exists in G_2 because it doesn't include any edge in $E_w \setminus z$. Compared with G_1, the cost of $P_{w'}$ reduces by $\sum_{e_i \in z} p_{e_i}^t - c_{e_i}$[6] in G_2. As to other paths in G_2,

[6] $p_{e_i} \geq c_{e_i}$ according to CCG-SET.

their costs reduce by $\sum_{e_i \in z} p^t_{e_i} - c_{e_i}$ at most, so $P_{w'}$ is also the shortest path in G_2. Note that G_2 is just the graph $G \backslash (E_w \backslash z)$, from the constraint (65), we have

$$\begin{aligned}\sum_{e_i \in E_w \backslash z} p_{e_i} &\le \sum_{e_i \in E_{w'} \backslash z} c_{e_i} \\ &= d_{G_2}(v_0, v_n) - \sum_{e_i \in z} c_{e_i} \\ &= d_{G \backslash (E_w \backslash z)}(v_0, v_n) - \sum_{e_i \in z} c_{e_i}\end{aligned} \tag{66}$$

Recall the constraint in $(C1)$ as

$$(C1):\ \sum_{e_i \in x} p_{e_i} \le d_{G \backslash x}(v_0, v_n) - (d_G(v_0, v_n) - \sum_{e_i \in x} c_{e_i}), \forall x \in E_w \tag{67}$$

Let $x = E_w \backslash z$, we have

$$\begin{aligned}\sum_{e_i \in E_w \backslash z} p_{e_i} &\le d_{G \backslash (E_w \backslash z)}(v_0, v_n) - \left(d_G(v_0, v_n) - \sum_{e_i \in (E_w \backslash z)} c_{e_i} \right) \\ &= d_{G \backslash (E_w \backslash z)}(v_0, v_n) - \sum_{e_i \in z} c_{e_i}\end{aligned} \tag{68}$$

The constraint (68) is $(C1)$ is just the same as the constraint (66), so the constraint we add into CCG-SET during each iteration is a standard constraint of $(C1)$. Then the constraint set CCG-SET is a subset of the constraint set $(C1)$. In each iteration, CCG-VCG algorithm adds a constraint of $(C1)$. The number of constraints in $(C1)$ is limited so that this algorithm must stop in a limited number of steps.

To prove the theorem, we just need to prove that the outcome of CCG-VCG algorithm is in the core. Assuming that the outcome of CCG-VCG algorithm isn't in the core. Thus, there is at least one constraint in $(C1)$ which is not satisfied by this result. Without loss of generality, let $x = E_w \backslash z'$ is the corresponding set, then the constraint becomes

$$\begin{aligned}\sum_{e_i \in E_w \backslash z'} p_{e_i} &> d_{G \backslash (E_w \backslash z')}(v_0, v_n) - \left(d_G(v_0, v_n) - \sum_{e_i \in (E_w \backslash z')} c_{e_i} \right) \\ &= d_{G \backslash (E_w \backslash z)}(v_0, v_n) - \sum_{e_i \in z} c_{e_i}\end{aligned} \tag{69}$$

Then we have

$$\begin{aligned}\sum_{e_i \in E_w} p_{e_i} &= \sum_{e_i \in E_w \backslash z'} p_{e_i} + \sum_{e_i \in z'} p_{e_i} \\ &> d_{G \backslash (E_w \backslash z)}(v_0, v_n) + \sum_{e_i \in z'} p_{e_i} - \sum_{e_i \in z'} c_{e_i}\end{aligned} \tag{70}$$

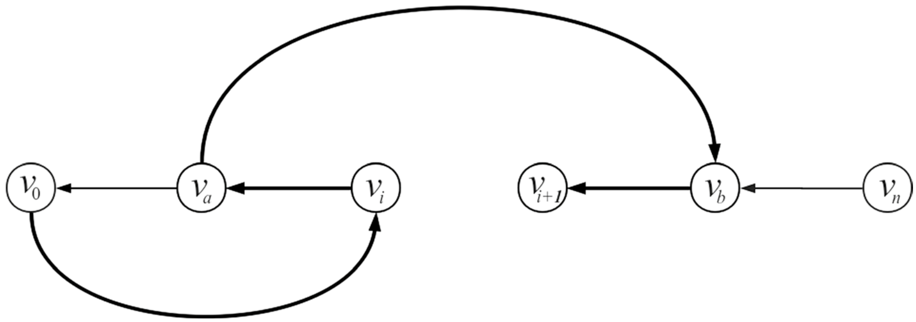

Fig. 12 Path $v_0 \rightarrow v_i \rightarrow v_a \rightarrow v_b \rightarrow v_{i+1}$

where $d_{G\backslash(E_w\backslash z)}(v_0, v_n)$ is the cost of the shortest path from v_0 to v_n in graph $G\backslash(E_w\backslash z)$. This path still exists in graph which changes the cost of edges in E_w from c_{e_i} to p_{e_i}. This change makes the cost of this path increase by $\sum_{e_i \in z'} p_{e_i} - \sum_{e_i \in z'} c_{e_i}$ at most. So the cost of this path is no more than $d_{G\backslash(E_w\backslash z)}(v_0, v_n) + \sum_{e_i \in z'} p_{e_i} - \sum_{e_i \in z'} c_{e_i}$, which means it is shorter than the sum of outcome in CCG-VCG algorithm. This produces a contradiction with terminal condition in CCG-VCG algorithm, so this theorem is established. □

Appendix C: Proof of Theorem 9

Proof In G', the edge in E_w is converted into a reverse edge with negative original cost. As we know, each edge is not cut edge for the connectivity from v_0 to v_n, that is, there exist a path from v_0 to v_n after removing this edge in graph G. We use the mathematical induction by proving the following two propositions.

1. There exist a path from v_0 to v_1 in G'.
2. If there exists a path from v_0 to v_i in G', then there exists a path from v_0 to v_{i+1} in G' $(0 < i < n)$.

It is obvious that Theorem 9 is established if these two propositions is correct. Note that $V_w(v_1, v_n)$ is the vertex set including $v_1, v_2, \dots, v_n$. In proposition 1, since (v_0, v_1) is not a cut edge, there must exist a path from v_0 to any vertex of $V_w(v_1, v_n)$ in the graph $G\backslash E_w$. Otherwise, there will not exist a path from v_0 to any vertex of $V_w(v_1, v_n)$ in graph $G\backslash(v_0, v_1)$, this is because compared with $G\backslash E_w$, the extra edges in $G\backslash(v_0, v_1)$ is useless for the connectivity between $\{v_0\}$ and $V_w(v_1, v_n)$. This means (v_0, v_1) is a cut edge, which is contradictory. Therefore, there exists a path from v_0 to any vertex of $V_w(v_1, v_n)$ in $G\backslash E_w$. This path also exists in G' and once this path could arrive at any vertex of $V_w(v_1, v_n)$ from v_0, it could arrive at v_1 along the negative edges in G'. Thus, proposition 1 is true.

In proposition 2, since there exists a path from v_0 to v_i in G', we can arrive at any vertex of $V_w(v_0, v_i)$ by just lengthening this path along the negative edges. Based on that (v_i, v_{i+1}) is not a cut edge, similarly, we can draw a conclusion that there exists a path from any vertex of $V_w(v_0, v_i)$ to any vertex of $V_w(v_{i+1}, v_n)$. Then there also exists a path from any vertex of $V_w(v_0, v_i)$ to any vertex of $V_w(v_{i+1}, v_n)$ in G'. Denote these two vertices by v_a, v_b and we have a path from v_0 to v_{i+1} as $v_0 \rightarrow v_i \rightarrow v_a \rightarrow v_b \rightarrow v_{i+1}$, like Fig. 12. Therefore, proposition 2 is proved.

Above all, Theorem 9 is established. □

References

1. Archer, A., & Tardos, É. (2007). Frugal path mechanisms. *ACM Transactions on Algorithms (TALG)*, *3*(1), 3.
2. Ausubel, L. M., & Milgrom, P. R. (2002). Ascending auctions with package bidding. *Advances in Theoretical Economics*, *1*(1), 1–42.
3. Ausubel, L. M., Milgrom, P., et al. (2006). The lovely but lonely Vickrey auction. *Combinatorial Auctions*, *17*, 22–26.
4. Bünz, B., Seuken, S., & Lubin, B. (2015). A faster core constraint generation algorithm for combinatorial auctions. In *Twenty-Ninth AAAI conference on artificial intelligence* (pp. 827–834).
5. Bünz, B., Lubin, B., & Seuken, S. (2018). Designing core-selecting payment rules: A computational search approach. In *Proceedings of the 2018 ACM conference on economics and computation* (pp. 109–109). ACM.
6. Cheng, H., Zhang, L., Zhang, Y., Wu, J., & Wang, C. (2018). Optimal constraint collection for core-selecting path mechanism. In *Proceedings of the 17th international conference on autonomous agents and multiagent systems* (pp. 41–49).
7. Clarke, E. H. (1971). Multipart pricing of public goods. *Public Choice*, *11*(1), 17–33.
8. Cramton, P. (2013). Spectrum auction design. *Review of Industrial Organization*, *42*(2), 161–190.
9. Day, R., & Milgrom, P. (2013). Optimal incentives in core-selecting auctions. In *The handbook of market design* (Chap. 11, pp. 282–298). OUP Oxford.
10. Day, R., & Milgrom, P. (2008). Core-selecting package auctions. *International Journal of Game Theory*, *36*(3–4), 393–407.
11. Day, R. W., & Cramton, P. (2012). Quadratic core-selecting payment rules for combinatorial auctions. *Operations Research*, *60*(3), 588–603.
12. Day, R. W., & Raghavan, S. (2007). Fair payments for efficient allocations in public sector combinatorial auctions. *Management Science*, *53*(9), 1389–1406.
13. Du, Y., Sami, R., & Shi, Y. (2010). Path auctions with multiple edge ownership. *Theoretical Computer Science*, *411*(1), 293–300.
14. Elkind, E., Sahai, A., & Steiglitz, K. (2004). Frugality in path auctions. In *Proceedings of the fifteenth annual ACM-SIAM symposium on discrete algorithms* (pp. 701–709). Society for Industrial and Applied Mathematics
15. Erdil, A., & Klemperer, P. (2010). A new payment rule for core-selecting package auctions. *Journal of the European Economic Association*, *8*(2–3), 537–547.
16. Feigenbaum, J., Papadimitriou, C., Sami, R., & Shenker, S. (2005). A BGP-based mechanism for lowest-cost routing. *Distributed Computing*, *18*(1), 61–72.
17. Grötschel, M., Lovász, L., & Schrijver, A. (1981). The ellipsoid method and its consequences in combinatorial optimization. *Combinatorica*, *1*(2), 169–197.
18. Groves, T., et al. (1973). Incentives in teams. *Econometrica*, *41*(4), 617–631.
19. Hartline, J., Immorlica, N., Khani, M.R., Lucier, B., & Niazadeh, R. (2018). Fast core pricing for rich advertising auctions. In *Proceedings of the 2018 ACM conference on economics and computation* (pp. 111–112). ACM.
20. Hershberger, J., & Suri, S. (2001). Vickrey prices and shortest paths: What is an edge worth? In *Proceedings 42nd IEEE symposium on foundations of computer science* (pp. 252–259). IEEE.
21. Karger, D., & Nikolova, E. (2006). On the expected VCG overpayment in large networks. In *Proceedings of the 45th IEEE conference on decision and control* (pp. 2831–2836).
22. Karlin, A.R., Kempe, D., & Tamir, T. (2005). Beyond VCG: Frugality of truthful mechanisms. In *46th annual IEEE symposium on foundations of computer science (FOCS'05)* (pp. 615–626). IEEE.
23. Lee, Y. T., Sidford, A., & Wong, S. C. W. (2015). A faster cutting plane method and its implications for combinatorial and convex optimization. In *56th annual symposium on foundations of computer science* (pp. 1049–1065). IEEE.
24. Lehmann, D., Oćallaghan, L. I., & Shoham, Y. (2002). Truth revelation in approximately efficient combinatorial auctions. *Journal of the ACM (JACM)*, *49*(5), 577–602.

25. Leskovec, J., & Krevl, A. (2014, June). SNAP datasets: Stanford large network dataset collection. http://snap.stanford.edu/data/.
26. Nisan, N., & Ronen, A. (1999). Algorithmic mechanism design. In *Proceedings of the thirty-first annual ACM symposium on theory of computing* (pp. 129–140). ACM.
27. Polymenakos, L., & Bertsekas, D. P. (1994). Parallel shortest path auction algorithms. *Parallel Computing*, *20*(9), 1221–1247.
28. Rothkopf, M. H., Pekeč, A., & Harstad, R. M. (1998). Computationally manageable combinational auctions. *Management science*, *44*(8), 1131–1147.
29. Vickrey, W. (1961). Counterspeculation, auctions, and competitive sealed tenders. *The Journal of Finance*, *16*(1), 8–37.
30. Yokoo, M., Sakurai, Y., & Matsubara, S. (2004). The effect of false-name bids in combinatorial auctions: New fraud in internet auctions. *Games and Economic Behavior*, *46*(1), 174–188.
31. Zhang, L., Chen, H., Wu, J., Wang, C. J., & Xie, J. (2016). False-name-proof mechanisms for path auctions in social networks. In *ECAI* (pp. 1485–1492).
32. Zhu, Y., Li, B., Fu, H., & Li, Z. (2014). Core-selecting secondary spectrum auctions. *IEEE Journal on Selected Areas in Communications*, *32*(11), 2268–2279.