Stabilizing Grand Cooperation of Machine Scheduling Game via Setup Cost Pricing

Lindong Liu

School of Management; International Institute of Finance

University of Science and Technology of China

Co-authored with Zikang Li (USTC)

NWPU-Online, July, 2020

Outline

Preliminaries

Cooperative Game

A **cooperative game** is defined by a pair (V, C):

- A set $V = \{1, 2, \ldots, v\}$ of players, **grand colaition**;
- A **characteristic function** $C(S) =$ the minimum total cost achieved by the cooperation of members in coalition $S \in \mathbb{S} = 2^V \setminus \{\emptyset\}$.

The game requires:

- A **cost allocation** $\alpha = [\alpha_1, \alpha_2, \ldots, \alpha_v] \in \mathbb{R}^v$, where $\alpha_k =$ the cost allocated to each player $k \in V$.

Core

Define $\alpha(S) = \sum_{k \in S} \alpha_k$.

A cost allocation $\alpha \in \mathbb{R}^v$ is in the **core** if it satisfies:

- **Budget Balance** Constraint: $\alpha(V) = C(V)$;
- **Coalition Stability** Constraints: $\alpha(S) \leq C(S)$ for each $S \in \mathbb{S}$.

$$\mathrm{Core}(V, C) = \Big\{\alpha : \ \alpha(V) = C(V), \\ \alpha(S) \leq C(S), \ \forall S \in \mathbb{S} \setminus \{V\}, \ \alpha \in \mathbb{R}^v\Big\}.$$

Core

Define $\alpha(S) = \sum_{k \in S} \alpha_k$.

A cost allocation $\alpha \in \mathbb{R}^V$ is in the **core** if it satisfies:

- **Budget Balance** Constraint: $\alpha(V) = C(V)$;
- **Coalition Stability** Constraints: $\alpha(S) \leq C(S)$ for each $S \in \mathbb{S}$.

$$\mathrm{Core}(V, C) = \left\{ \alpha : \ \alpha(V) = C(V), \right.$$
$$\left. \alpha(S) \leq C(S), \ \forall S \in \mathbb{S} \setminus \{V\}, \ \alpha \in \mathbb{R}^V \right\}.$$

However, $\mathrm{Core}(V, C)$ can be empty.

Existing Instruments

$$\mathrm{Core}(V, C) = \left\{\alpha : \ \alpha(V) = C(V), \ \alpha(S) \le C(S), \ \forall S \in \mathbb{S} \setminus \{V\}\right\}$$

Existing Instruments

$$\mathrm{Core}(V, C) = \left\{ \alpha : \ \alpha(V) = C(V), \ \alpha(S) \leq C(S), \ \forall S \in \mathbb{S} \setminus \{V\} \right\}$$

- Subsidization: $\alpha(V) = C(V) - \theta$, $\epsilon-$**core**;

Existing Instruments

$$\mathrm{Core}(V, C) = \left\{ \alpha : \ \alpha(V) = C(V), \ \alpha(S) \leq C(S), \ \forall S \in \mathbb{S} \setminus \{V\} \right\}$$

- Subsidization: $\alpha(V) = C(V) - \theta$, $\epsilon-$**core**;
- Penalization: $\alpha(S) \leq C(S) + z$, **least core**;

Existing Instruments

$$\mathrm{Core}(V, C) = \left\{ \alpha : \ \alpha(V) = C(V), \ \alpha(S) \le C(S), \ \forall S \in \mathbb{S} \setminus \{V\} \right\}$$

- Subsidization: $\alpha(V) = C(V) - \theta$, $\epsilon-$**core**;
- Penalization: $\alpha(S) \le C(S) + z$, **least core**;
- Simul. S & P: $\alpha(V) = C(V) - \theta$ and $\alpha(S) \le C(S) + z$, **PSF**;

Existing Instruments

$$\mathrm{Core}(V, C) = \left\{ \alpha : \ \alpha(V) = C(V), \ \alpha(S) \leq C(S), \ \forall S \in \mathbb{S} \setminus \{V\} \right\}$$

- Subsidization: $\alpha(V) = C(V) - \theta$, $\epsilon-$**core**;
- Penalization: $\alpha(S) \leq C(S) + z$, **least core**;
- Simul. S & P: $\alpha(V) = C(V) - \theta$ and $\alpha(S) \leq C(S) + z$, **PSF**;
- Inv. Opt.: Changing c to d such that $\mathrm{Core}(V, D)$ is non-empty.

Existing Instruments

$$\mathrm{Core}(V, C) = \left\{\alpha : \ \alpha(V) = C(V), \ \alpha(S) \le C(S), \ \forall S \in \mathbb{S} \setminus \{V\}\right\}$$

- Subsidization: $\alpha(V) = C(V) - \theta$, $\epsilon-$**core**;
- Penalization: $\alpha(S) \le C(S) + z$, **least core**;
- Simul. S & P: $\alpha(V) = C(V) - \theta$ and $\alpha(S) \le C(S) + z$, **PSF**;
- Inv. Opt.: Changing c to d such that $\mathrm{Core}(V, D)$ is non-empty.

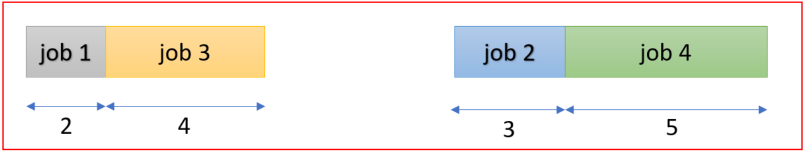
S.	Caprara and Letchford (2010, MP), Liu et al. (2016, IJOC)
P.	Faigle et al. (2001, IJGT), Schulz and Uhan (2010, OR)
P&S	Liu et al. (2018, OR)
Inv. Opt.	Liu et al. (2020, under review)

Illustrative Example

Example: Machine Scheduling Game (MSG)

Game of Parallel Machine Scheduling with Setup Cost:

- Grand coalition: $V = \{1, 2, 3, 4\}$;
- Processing times: $t_1 = 2,\ t_2 = 3,\ t_3 = 4,\ t_4 = 5$;
- Machine setup cost: $t_0 = 9.5$;
- $C(S)$ for $S \in \mathbb{S}$: minimizing the total completion time of jobs in S plus the machine setup cost;
- $C(V) = C(\{1,3\}) + C(\{2,4\}) = 8 + 11 + 9.5 \times 2 = 38.$



Example: Empty Core

Coalitions	Cost
$\{1\}$	11.5
$\{2\}$	12.5
$\{3\}$	13.5
$\{4\}$	14.5
$\{1,2\}$	16.5
$\{1,3\}$	17.5
$\{1,4\}$	18.5
$\{2,3\}$	19.5
$\{2,4\}$	20.5
$\{3,4\}$	22.5
$\{1,2,3\}$	25.5
$\{1,2,4\}$	26.5
$\{1,3,4\}$	28.5
$\{2,3,4\}$	31.5
$\{1,2,3,4\}$	38

Example: Empty Core

Coalitions	Cost
$\{1\}$	11.5
$\{2\}$	12.5
$\{3\}$	13.5
$\{4\}$	14.5
$\{1,2\}$	16.5
$\{1,3\}$	17.5
$\{1,4\}$	18.5
$\{2,3\}$	19.5
$\{2,4\}$	20.5
$\{3,4\}$	22.5
$\{1,2,3\}$	25.5
$\{1,2,4\}$	26.5
$\{1,3,4\}$	28.5
$\{2,3,4\}$	31.5
$\{1,2,3,4\}$	38

Optimal Cost Allocation Problem

$$\begin{aligned} \max\ & \left(\alpha_1+\alpha_2+\alpha_3+\alpha_4\right) = 37.25 < 38 \\ s.t.\ & \alpha_1 \leq 11.5,\ \cdots,\ \alpha_4 \leq 14.5, \\ & \alpha_1+\alpha_2 \leq 16.5,\ \cdots,\ \alpha_3+\alpha_4 \leq 22.5, \\ & \cdots, \\ & \alpha_1+\alpha_2+\alpha_3+\alpha_4 \leq 38. \end{aligned}$$

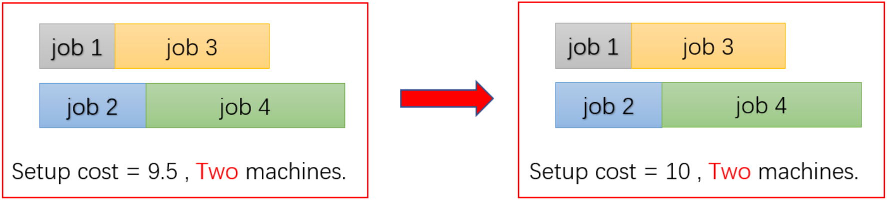
$$\alpha^* = \left[6; 8.75; 10.75; 11.75\right]$$

The minimum subsidy:

$$C(V) - \alpha(V) = 38 - 37.25 = 0.75$$

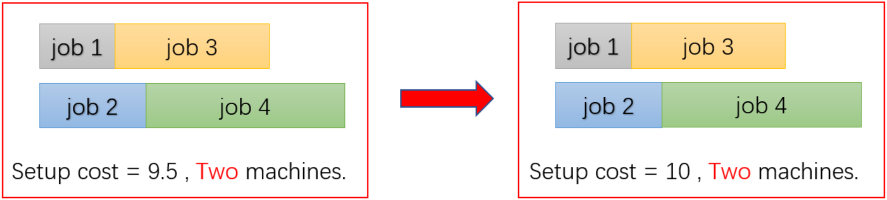
Example: Pricing Instrument

Increase the setup cost from 9.5 to 10.



Example: Pricing Instrument

Increase the setup cost from 9.5 to 10.

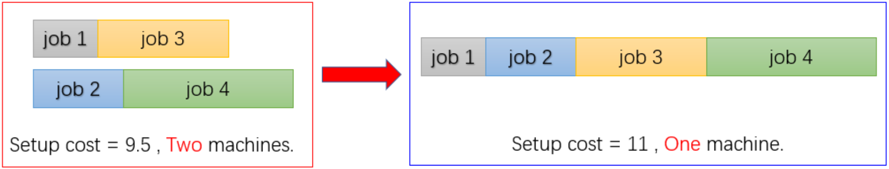


Setup cost	Increment	Num of Machines	Total pricing	C(V)	α(V)	Subsidy
9.5	0	2	0	38	37.25	0.75
10	0.5	2	1	39	38	1

The total pricing can exactly cover the gap, which means the grand coalition can be stabilized by the players themselves.

Example: Pricing Instrument

Increase the setup cost from 9.5 to 11.14.
For the grand coalition, it only needs one machine now.



Setup cost	Increment	Num of Machines	Total pricing	C(V)	α(V)	Subsidy
9.5	0	2	0	38	37.25	0.75
11.14	1.64	1	1.64	41.14	39.5	1.64

Models & Analyses

Problem Definition and Formulation

Definition

Parallel Machine Scheduling Game with Setup Cost (MSG):

Problem Definition and Formulation

Definition

Parallel Machine Scheduling Game with Setup Cost (MSG):

- Grand coalition: $V = \{1, 2, \ldots, v\}$;

Problem Definition and Formulation

Definition

Parallel Machine Scheduling Game with Setup Cost (MSG):

- Grand coalition: $V = \{1, 2, \ldots, v\}$;
- Identical machines: $M = \{1, 2, \ldots, m\}$;

Problem Definition and Formulation

Definition

Parallel Machine Scheduling Game with Setup Cost (MSG):

- Grand coalition: $V = \{1, 2, \ldots, v\}$;
- Identical machines: $M = \{1, 2, \ldots, m\}$;
- Setup cost (unit price) of opening every single machine: P;

Problem Definition and Formulation

Definition

Parallel Machine Scheduling Game with Setup Cost (MSG):

- Grand coalition: $V = \{1, 2, \ldots, v\}$;
- Identical machines: $M = \{1, 2, \ldots, m\}$;
- Setup cost (unit price) of opening every single machine: P;
- Processing time of each job: $t_k, \ \forall k \in V$;

Problem Definition and Formulation

Definition

Parallel Machine Scheduling Game with Setup Cost (MSG):

- Grand coalition: $V = \{1, 2, \ldots, v\}$;
- Identical machines: $M = \{1, 2, \ldots, m\}$;
- Setup cost (unit price) of opening every single machine: P;
- Processing time of each job: t_k, $\forall k \in V$;
- Characteristic function: $C(S)$, denoting the total completion time and setup cost for each coalition $S \in \mathbb{S}$.

Problem Definition and Formulation

Definition

The characteristic function value, $C(S)$, of MSG is given by ILP

$$C(S,P) = \min \sum_{k \in V} \sum_{j \in O} c_{kj} x_{kj} + P \sum_{k \in S} x_{k1}$$

$$\begin{aligned}
s.t. \quad & \sum_{j \in O} x_{kj} - y_k^S = 0, \forall k \in V, \\
& \sum_{k \in V} x_{kj} \le m, \forall j \in O, \\
& x_{kj} \in \{0,1\}, \forall k \in V, \forall j \in O, \\
& y_k^S = 1, k \in S \,; y_k^S = 0, k \notin S.
\end{aligned}$$

Properties

Definition

- $\left[P_L(i,S), P_H(i,S)\right]$: Price range of using i machines for scheduling jobs among players S;

Properties

Definition

- $\left[P_L(i,S), P_H(i,S)\right]$: Price range of using i machines for scheduling jobs among players S;
- $\left[0, P^*\right]$: Effective domain of pricing MSG for stabilization;

Properties

Definition

- $\left[P_L(i,S), P_H(i,S)\right]$: Price range of using i machines for scheduling jobs among players S;
- $\left[0, P^*\right]$: Effective domain of pricing MSG for stabilization;
- P_i: For easy of exposition, let $P_1 = P^*$ and $P_i = P_H(i,V) = P_L(i-1,V)$. Thus, the effective domain of $\left[0, P^*\right]$ is divided into v non-overlapping sub-intervals by P_i, $\forall i = \{2, 3, ..., v\}$.

Note: P^* is the lowest price under which the grand cooperation of MSG is stable and it uses only one machine in the optimal scheduling decision, i.e., $P^* \in \left[P_L(1,V), P_H(1,V)\right]$ and MSG $\left(V, C(\ \cdot\ , P^*)\right)$ has non-empty core.

Properties

Gap: $\omega(P) \quad = \min_{\alpha}\{C(V,P) - \alpha(V) : \alpha(S) \leq C(S,P), \forall S \in \mathbb{S}\}$

Theorem 1

Function $\omega(P)$ is piecewise linear, and convex in P at each sub-interval $[P_{i+1}, P_i]$.

Lemma 1

Breaking price, P_i $(2 \leq i \leq v)$, can be obtained by SPT rules in polynomial time.

Theorem 2

Boundary price P^* equals $\sum_{i=2}^{v} P_i$, and can be obtained in polynomial time.

Theorem 3

Gap $\omega(P) = 0$ when the number of machines used by $C(V,P)$ is larger than $\frac{v}{2}$.

Properties

Single Machine Scheduling Game (SMSG):

$$C(S, P, \text{``}m = \text{''}1) = \min \sum_{k\in V} \sum_{j\in O} c_{kj}x_{kj} + P$$

$$s.t. \quad \sum_{j\in O} x_{kj} - y_k^S = 0, \ \forall k \in V,$$

$$\sum_{k\in V} x_{kj} \leq 1, \ \forall j \in O,$$

$$x_{kj} \in \{0, 1\}, \ \forall k \in V, \ \forall j \in O, \ y_k^S = 1, \ \forall k \in S\,; y_k^S = 0, \ \forall k \notin S.$$

Theorem 4

For SMSG (or, MSG with only one machine), given $P \in [0, P^*]$ (or, $[P_2, P_1]$), the slope of each segment for function $\omega(P)$ is within $\left(-1, -\frac{1}{v-1}\right]$, and the number of breakpoints for function $\omega(P)$ is within $O(v^2)$.

Properties

Definition

Define that

$$\omega_1(P) = \min_{\alpha} \left\{ C(V, P) - \alpha(V) : \alpha(S) \leq C(S, P, 1),\ \forall S \in \mathbb{S} \setminus \{V\} \right\}$$

Theorem 5

The original problem $\omega(P)$ is equivalent to $\omega_1(P)$, which is polynomially solvable by cutting plane (see CP Algorithm below).

Algorithms & Computations

CP Algorithm

Cutting Plane Algorithm for Computing $\omega_1(P)$ under Given P

Step 1. Let $\mathbb{S}' \subseteq \mathbb{S} \backslash \{V\}$ indicates a restricted coalition set, which includes some initial coalitions, e.g.,$\{1\}, \{2\}, \ldots, \{v\}$.

Step 2. Find an optimal solution $\bar{\alpha}$ to LP

$$\max_{\alpha \in \mathbb{R}^n} \left\{ \alpha(V) : \alpha(S) \leq C(S, P, 1), \text{ for all } S \in \mathbb{S}' \right\}.$$

Step 3. Find an optimal solution S^* to separation problem

$$\delta = \min \left\{ C(S, P, 1) - \bar{\alpha}(S) : \forall S \in \mathbb{S} \setminus \{V\} \right\}.$$

Step 4. If $\delta < 0$, then add S^* to $\mathbb{S}'$, and go to step 2; otherwise, return $\omega(P) = \omega_1(P) = C(V, P) - \bar{\alpha}(V)$.

DP Algorithm

Dynamic Programming for Solving Separation Problem

- Without loss of generality, assuming $t_1 \geq t_2 \geq \ldots \geq t_v$.
- Recall that For the separation problem is given by

$$\delta = \min \left\{ C(S, P, 1) - \bar{\alpha}(S) : \forall S \in \mathbb{S} \setminus \{V\} \right\}.$$

- Note: If some player $k \notin S$ is added into S, where $|S| = u$, the increment of δ is $(u+1)t_k - \alpha_k$ (see recursion in Step 3).

Step 1. Initially, let $D(k, u)$ indicate the minimum objective value of the restricted problem of separation problem δ, where k is a player in the grand coalition and u is the number of players included in S. So $k \in \{1, 2, \ldots, v\}$ and $u \in \{0, 1, \ldots, v\}$.

DP Algorithm

Step 2. Given the initial conditions $D(1,0) = P$ and $D(1,1) = t_1 - \alpha_1 + P$. The boundary conditions are $D(k,u) = \infty$ if $u > k$, for all $k \in V$.

Step 3. Given the recursion:

$$D(k,u) = \min \begin{cases} D(k-1,u), \text{when } S^* \text{ does not contain } k, \\ D(k-1,u-1) + ut_k - \alpha_k, \text{when } S^* \text{ contains } k. \end{cases}$$

Step 4. Obtain the optimal objective value of the separation problem by $\delta = \min\{D(v,u) : u \in \{1,2,\ldots,v-1\}\}$. Return δ.

DP Algorithm can solve the separation problem δ in $O(v^2)$ time.

Note that $k \in \{1,2,\ldots,v\}$ and $u \in \{0,1,\ldots,v\}$ in $D(k,u)$, thus the time complexity of this DP is $O(v^2)$.

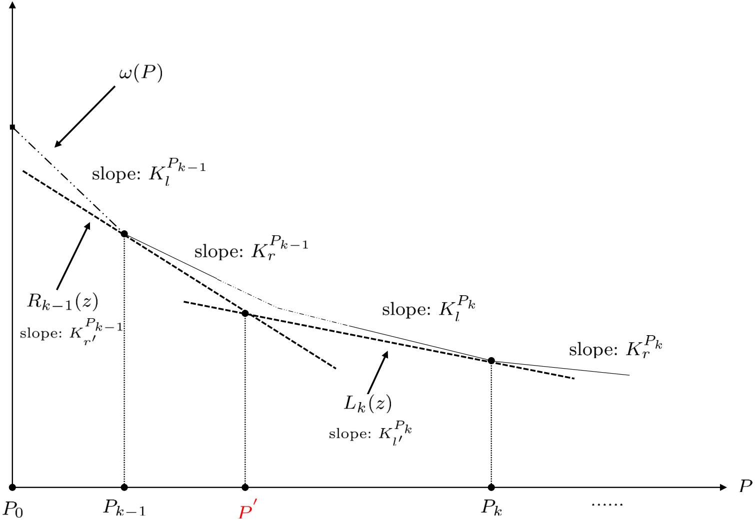
IPC Algorithm

DP (P-time) $\Rightarrow$ CP (P-time) $\Rightarrow$ $\omega(P)$ for given P (P-time)

Intersection Points Computation for Constructing Function $\omega(P)$

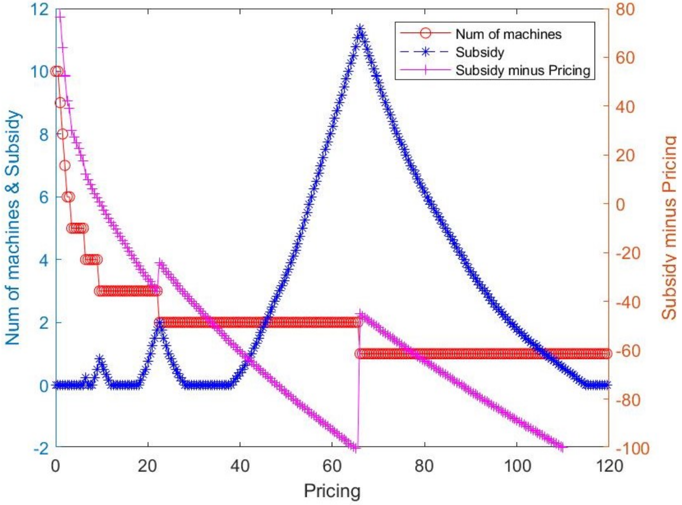
Since function $\omega(P)$ is shown to be piecewise linear, convex and have polynomial number of breakpoints within each interval $[P_{i+1}, P_i]$ ($i = v-1, v-2, ..., 1$), we claim that function $\omega(P)$ can be polynomially constructed by the IPC algorithm developed in Liu et.al. 2018 (OR).

IPC Algorithm



Computational Results

Processing time:[1, 1.5, 2, 2.5, 3.5, 4, 4, 6.5, 6.5, 7]



Extension & Generalization

Machine Scheduling Game with Weighted Jobs

Definition

Machine Scheduling Game with Weighted Jobs (WSGW):

- Each job $k \in V$ has a processing time, t_k, and a weight, w_k.

Properties

- $C(S, P)$ and P_i $(2 \leq i \leq v)$ can be obtained by analysing the order of t_k/ω_k instead of t_k.
- $\omega(P)$ is piecewise linear, convex in price P at each subinterval.
- IPC, CP Algorithms can be used to construct function $\omega(P)$ in polynomial time.

Pricing in General IM Games

Definition

General Integer Minimization Games:

- $C(S,P) = \min_x\{cx + Pm(x) : Ax \geq By^S + D, \beta x \leq m, x \in \mathbb{Z}^t\}$
- Let $H_i = C(V,P) - Pi$, where $i = m(x^*)$ and x^* solves $C(V,P)$.

Properties for MSG

- $H_{i-1} - H_i > 0 \Leftrightarrow P_i > 0, \ \forall i = 2, 3, \ldots, v.$
- $H_i - H_{i+1} < H_{i-1} - H_i \Leftrightarrow P_i > P_{i+1}, \ \forall i = 2, 3, \ldots, v-1.$
- Intervals $[P_v, P_{v-1}]$, $[P_{v-1}, P_{v-2}]$, ..., $[P_2, P_1]$, $[P_1, P^*]$ are non-overlapping.

Conclusions

⋆ Cooperative Game Theory:

– Stabilizing Grand Cooperation via Pricing.

⋆ Scheduling Problem:

– Parallel Machine Scheduling with Setup Cost.

⋆ Models, Solution Methods and Applications:

– Several ILP formulations;

– CP, DP and IPC algorithms for constructing function $\omega(P)$.

The End

Thank you!