

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 (PG Student, USTC)

NWPU-Online, July, 2020

Outline

- 1 Preliminaries
- 2 Motivation and Illustrative Example
- 3 Models and Analyses
- 4 Algorithms and Computations
- 5 Extension and Generalization
- 6 Conclusion

PRELIMINARIES

Cooperative Game

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

- A set $V = \{1, 2, \dots, v\}$ of players, **grand coalition**;
- 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** $\alpha = [\alpha_1, \alpha_2, \dots, \alpha_v] \in \mathbb{R}^v$, where α_k = the cost allocated to each player $k \in V$.

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 $S \in \mathbb{S}$.

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

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 $S \in \mathcal{S}$.

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

However, $\text{Core}(V, c)$ can be empty.

Existing Instruments

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

Existing Instruments

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

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

Existing Instruments

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

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

Existing Instruments

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

- Subsidization: $\alpha(V) = C(V) - \theta$, ϵ -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;

Existing Instruments

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

- Subsidization: $\alpha(V) = C(V) - \theta$, ϵ -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 $\text{Core}(V, D)$ is non-empty.

Existing Instruments

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

- Subsidization: $\alpha(V) = C(V) - \theta$, ϵ -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 $\text{Core}(V, D)$ 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. \(2018, OR\)](#)

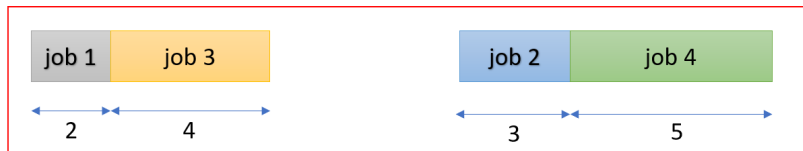
Inv. Opt. [Liu et al. \(2020, under review\)](#)

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 $S \in \mathbb{S}$: minimizes the total completion time of jobs in S plus the machine setup cost;
- $\pi(N) = \pi(\{1, 3\}) + \pi(\{2, 4\}) = 38$ (SPT Rule).



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

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

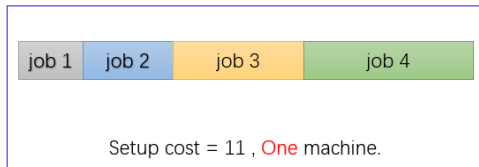
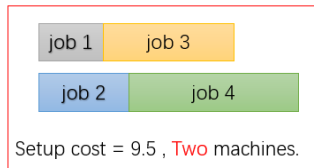
$$\alpha^* = [6; 8.75; 10.75; 11.75]$$

The minimum subsidy:

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

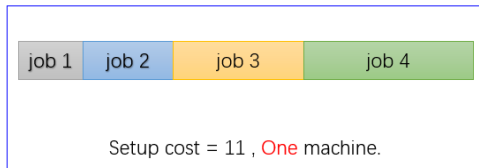
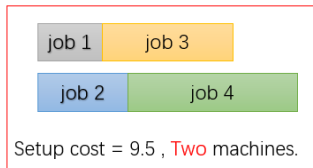
Example: Pricing Instrument

Increase the setup cost from 9.5 to 11.
For the grand coalition, it only needs one machine.



Example: Pricing Instrument

Increase the setup cost from 9.5 to 11.
For the grand coalition, it only needs one machine.



Setup cost	Increment	Num of Machines	Total pricing	$c(V)$	$\alpha(V)$	Subsidy
9.5	0	2	0	38	37.25	0.75
10	0.5	2	1	39	38	1
11.14	1.64	1	1.64	41.14	39.5	1.64

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

MODELS & ANALYSES

Problem Definition and Formulation

Definition

An Identical Variable Parallel machine scheduling of Unweighted jobs game (IVPU):

Problem Definition and Formulation

Definition

An Identical Variable Parallel machine scheduling of Unweighted jobs game (IVPU):

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

Problem Definition and Formulation

Definition

An Identical Variable Parallel machine scheduling of Unweighted jobs game (IVPU):

- **Grand coalition:** $V = \{1, 2, \dots, v\}$;
- **Identical machine:** $M = \{1, 2, \dots, m\}$;

Problem Definition and Formulation

Definition

An Identical Variable Parallel machine scheduling of Unweighted jobs game (IVPU):

- **Grand coalition:** $V = \{1, 2, \dots, v\}$;
- **Identical machine:** $M = \{1, 2, \dots, m\}$;
- **Each machine Price:** P and **Each job processing time:** t_k ;

Problem Definition and Formulation

Definition

An Identical Variable Parallel machine scheduling of Unweighted jobs game (IVPU):

- **Grand coalition:** $V = \{1, 2, \dots, v\}$;
- **Identical machine:** $M = \{1, 2, \dots, m\}$;
- **Each machine Price:** P and **Each job processing time:** t_k ;
- **Characteristic function:** $c(S) = \min(\sum_{k \in S} C_k + Pm_S)$,

where C_k is the completion time of job $k \in S$ and m_S is the number of using machine for the sub-coalition S .

Problem Definition and Formulation

Definition

A cooperative TU game (V, c) is called an IVPU game if it satisfies the following formulations:

Problem Definition and Formulation

Definition

A cooperative TU game (V, c) is called an IVPU game if it satisfies the following formulations:

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

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

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

$$x_{kj} \in \{0, 1\}, \forall k \in V, \forall j \in O,$$

$$y_k^S = 1, k \in S; y_k^S = 0, k \notin S.$$

Definition

- The interval $[P_L(m, S), P_H(m, S)]$ denotes the value range of price when the number of using machines m and the coalition S are given.

Definition

- The interval $[P_L(m, S), P_H(m, S)]$ denotes the value range of price when the number of using machines m and the coalition S are given.
- For the grand coalition V , define $P_L(v, V) = 0, P_H(1, V) = P^*$. So the practical domain of the price is $[0, P^*]$ which is divided into v nonoverlapping subintervals by the number of using machines.

Definition

- The interval $[P_L(m, S), P_H(m, S)]$ denotes the value range of price when the number of using machines m and the coalition S are given.
- For the grand coalition V , define $P_L(v, V) = 0, P_H(1, V) = P^*$. So the practical domain of the price is $[0, P^*]$ which is divided into v nonoverlapping subintervals by the number of using machines.
- Denote the right end of every subinterval as $P_i, 1 \leq i \leq v$, where $P_1 = P^*$.

Properties

$$\omega(P) = \min_{\alpha} \{c(V, m(V, P)) - \alpha(V) : \\ \alpha(s) \leq c(s, m(s, P)), \forall s \in S, \alpha \in \mathbb{R}^v\};$$

Theorem 1

$\omega(P)$ is piecewise linear, and convex in price P at each subinterval $[P_L(m, V), P_H(m, V)]$ in $[0, P^*]$.

Lemma 1

$P_i, 2 \leq i \leq v$ can be obtained by SPT rules.

Theorem 2

$$P_1 = P_2 + \cdots + P_n = \sum_{i=2}^n P_i.$$

Theorem 3

$\omega(P)$ can be bounded by zero when the number of using machines, m , is larger than $\frac{n}{2}$.

Properties

Theorem 4

When the number of using machines is 1 for the grand coalition, the range of slopes of the line segments in the interval is $(-1, -\frac{1}{n-1}]$, and the number of breakpoints is $O(v^2)$.

Theorem 5

Define that

$$\omega_1(P) = \min_{\alpha} \{c(V, m(V, P)) - \alpha(V) : \\ \alpha(s) \leq c(s) + P, \forall s \in S, \alpha \in \mathbb{R}^v\}$$

Then the original problem $\omega(P)$ is equivalent to $\omega_1(P)$ which means that all sub-coalitions only use **one** machine.

ALGORITHMS & COMPUTATIONS

IPC Algorithm

The Intersection Points Computation Algorithm to Construct $\omega(P)$ Function.

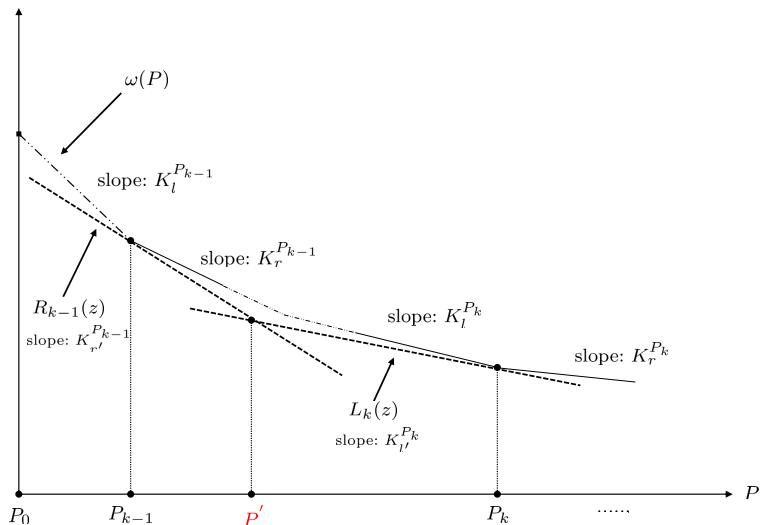
- Step 1.** Initially, set $I^* = \{P_L, P_H\}$ and $\mathbb{I} = \{[P_L, P_H]\}$.
- Step 2.** If \mathbb{I} is not empty, update I^* and \mathbb{I} by the following steps:
- Step 3.** Sort values in I^* by $P_0 < P_1 < \dots < P_q$, where $P_0 = P_L, P_q = P_H$ and $q = |I^*| - 1$.
- Step 4.** Select any interval from \mathbb{I} , denoted by $[P_{k-1}, P_k]$ with $1 \leq k \leq q$.
- Step 5.** Construct two linear function $R_{k-1}(P)$ and $L_k(P)$ so that $R_{k-1}(P)$ passes $(P_{k-1}, \omega(P_{k-1}))$ with a slope equal to a right derivative $K_r^{P_{k-1}}$ of $\omega(P)$ at P_{k-1} , and that $L_k(z)$ passes $(P_k, \omega(P_k))$ with a slope equal to a left derivative $K_l^{P_k}$ of $\omega(P)$ at P_k .

Step 6. If $R_{k-1}(P)$ passes $(P_k, \omega(P_k))$ or $L_k(P)$ passes $(P_{k-1}, \omega(P_{k-1}))$, then update \mathbb{I} by removing $[P_{k-1}, P_k]$. Otherwise, $R_{k-1}(P)$ and $L_k(P)$ must have a unique intersection point at $P = P'$ for some $P' \in (P_{k-1}, P_k)$. Update I^* by adding P' , and update \mathbb{I} by removing $[P_{k-1}, P_k]$, adding $[P_l, P']$ and $[P', P_r]$.

Step 7. Go to step 2.

Step 8. Return a piecewise linear function by connecting points $(P, \omega(P))$ for all $P \in I^*$.

IPC Algorithm



The Cutting Plane Algorithm to compute $\omega(P)$ for a given P .

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

Step 2. Find an optimal solution $\bar{\alpha}(\cdot, P)$ to LP $\tau(P)$:

$$\max_{\alpha \in \mathbb{R}^n} \{ \alpha(N, P) : \alpha(s, P) \leq c(s) + P, \text{ for all } s \in \mathbb{S}' \}.$$

Step 3. Find an optimal solution s^* to **the separation problem**:

$$\delta = \min \{ c(s) + P - \bar{\alpha}(s, z) : \forall s \in \mathbb{S} \setminus \{N\} \}.$$

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

DP Algorithm

The Dynamic Programming Algorithm to solve the separation problem.

Step 1. Initially, let $D(k, u)$ indicate the minimum objective value of the restricted problem of separation problem, where $k \in \{1, 2, \dots, v\}$ and $u \in \{0, 1, \dots, v\}$.

Step 2. Given the initial conditions $D(1, 0) = P$ and $D(1, 1) = t_1 - \beta_1 + P$. The boundary conditions are $D(k, u) = \infty$ if $u > k$, for all $k \in V$.

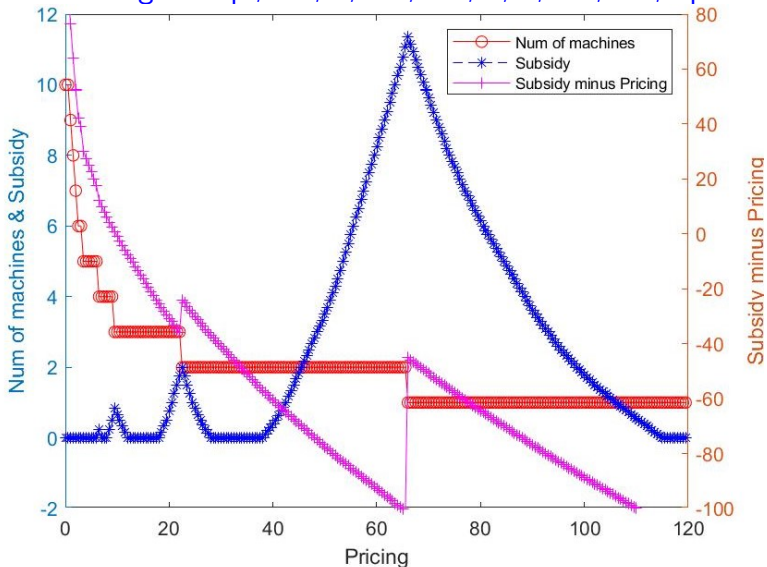
Step 3. Given the recursion:

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

Step 4. Obtain the optimal objective value of separation problem by $\delta_{AIPU} = \min\{D(v, u) : u \in \{1, 2, \dots, v-1\}\}$. return δ_{AIPU} .

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

A Machine Scheduling Game with Weighted Jobs:

- Each job $k \in V$ has a processing time and a weight denoted by t_k, ω_k , respectively.
- $c(S)$ can be obtained by assuming that
$$t_1/\omega_1 \leq t_2/\omega_2 \leq \dots \leq t_v/\omega_v.$$
- $P_m = c_0(V, m) - c_0(V, m - 1), 2 \leq m \leq v.$
- The properties of $\omega(P)$ and the corresponding Algorithms are similar to unweighted one.

Pricing in General IM Games

Definition

The General Integer Minimization Games:

- **ILP**: $c(S, m(S, P)) = \min_x \{cx + Pm(x) : Ax \geq By^s + D, \tilde{\alpha}x \leq m, x \in \mathbb{Z}^{t \times 1}\}$
- **Decompose** $c(S, m(S, P))$ into $c_0(S, m(S)) + Pm$.
- Analyse the properties of $c_0(S, m(S))$.

CONCLUSIONS

- ★ **Cooperative Game Theory:**
 - New Instrument for Stabilization via Setup cost Pricing.
- ★ **Scheduling Problem:**
 - Parallel Machine Scheduling with Setup Cost.
- ★ **Models, Solution Methods and Applications:**
 - Several ILP formulations;
 - Cutting Plane to solve the separation problem;
 - Implementations on the MSGW game.

Thank you!