Smart-CPR: Self-Organisation and Self-Governance in the Sharing Economy

David Kurka Jeremy Pitt

SASO^ST 2017, Tucson, AZ, September 22, 2017

Imperial College London

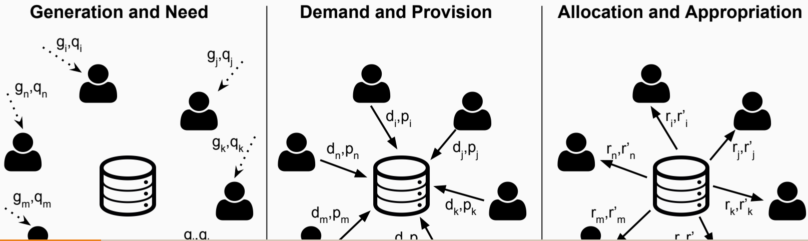
Introduction

Motivation

- The increasing automation and capacity of communication of **industrial systems** brings new possibilities and challenges to the sector.
- We investigate a problem of distributed and collective supply and discuss solutions to the issue of **fair and reliable** decision making in open systems.
- By combining principles of **social organisation and cooperation** with **blockchain and smart-contract** technologies, we can develop a system able to:
    - intermediate communication between producers;
    - make justifiable decisions of resource allocation; and
    - avoid and punish possible abuses from the participants.

Problem Formulation

- Producers independently operate in a market of common manufactured widgets
- Over time, producers receive demands for products (q) and keep a capacity of production output (g)
- Producers cooperate by distributing offer (p) and demand (d) in a common pool of services
- Resources of industrial supply and demand are distributed (r) among producers, following stipulated rules



Questions

- How to ensure **efficient, fair, inclusive and sustainable** ways to distribute services among producers?
- What **rules and norms** should govern the interactions and transactions?
- **Who** should determine the stipulated rules?
- How to deal with **abuses** and **non-compliance** to the rules?

System Design and Algorithm

Actors Model: Smart-CPR

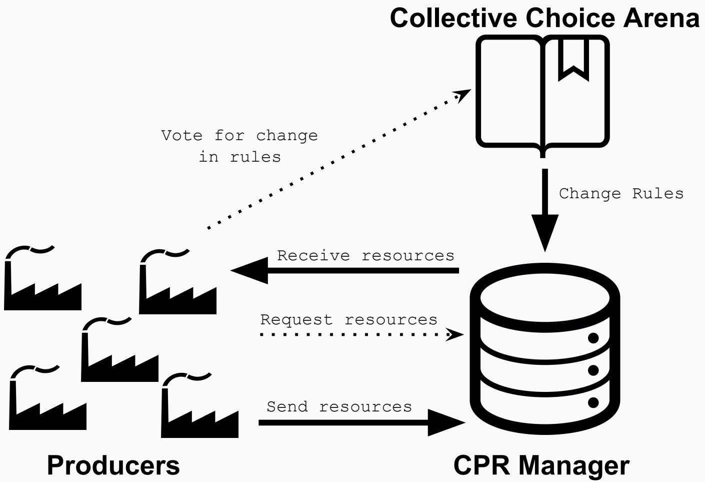


Figure 2: Smart-CPR system model with interactions between the system's actors.

Producer

Algorithm 1 Producer

Attributes:

$Need$ - resources currently needed

$Avail$ - resources available

$PKeep$ - prob of user not sharing available resource

$PoolAnswers$ - record of CPR Manager's answers to requests

event ResourceGeneration(G)

  $Decision \leftarrow$ KeepOrContribute$(PKeep)$

  if $Decision = keep$ **then**

    $Provision \leftarrow \text{rand}(0, 1) * G$

  else if $Decision = contribute$ **then**

    $Provision \leftarrow G$

  end if

  Send($CPRManager, G$)

  $Avail \leftarrow G - Provision$

- Responsible for mediating provision and demands of resources to the CPR Manager

Algorithm 2 CPR Manager

Attributes:

$Pool$ - amount of resources available in the common-pool

$Rules$ - smart-contract, specifying the rules used to judge whether a request should be accepted or not

$Ledger$ - blockchain recording users' interaction with CPR Manager

event RECEIVEPROVISION(P, $UserId$)

$Pool \leftarrow Pool + P$

$Ledger \leftarrow$ UPDATELEDGER($UserId$, provision, P)

end event

event RESOURCEREQUEST(D, $UserId$)

$Ledger \leftarrow$

- Smart-Contract define policy and rules for responses for resources request
- Responsible for fast decision making of resource allocation

Arena for Collective Choice

Algorithm 3 Collective Choice Arena

Attributes:

$VotesHistory$- record of users' past votes, used to define new policies

$CurrentRules$ - smart-contract that is currently being implemented by CPR Manager

$Threshold$ - number of votes considered to trigger change

event ReceiveVote(V, $UserId$)

$VotesHistory \leftarrow VotesHistory + V$

$ReceivedVotes \leftarrow$ length($VotesHistory$)

if $ReceivedVotes \geq Threshold$ **then**

$NewRules \leftarrow$ NewSmartContract($VotesHistory$)

Send($SmartPool$, $NewRules$)

else if $ReceivedVotes < Threshold$

- Compute producers' votes for change on CPR Manager rules
- Issue new smart-contracts to be used as policy for CPR Manager

Policy Making - Smart-Contract

Rescher's legitimate claims of justice as used as metric to evaluate producer participation in the system (according to the public ledger)

Canons of equality	$\phi_i^1 = R_i$
	$\phi_i^2 = \begin{cases} (1-\alpha)\cdot\phi_i^2 + \alpha & \text{if accepted req.} \\ (1-\beta)\cdot\phi_i^2 & \text{if denied req.} \end{cases}$
Canon of needs	$\phi_i^3 = D_i$
Canon of productivity	$\phi_i^4 = P_i$
Canon of effort	$\phi_i^5 = CurTime - JoiningTime$
Canon of social utility	$\phi_i^6 = Status(i)$

Upon a request for resource, a weighted sum of the claims is computed and a response is evaluated based on a smart-contract policy

Algorithm Smart-Contract

Require:

$W = [w_1, w_2 \dots w_n]$ ▷ Weights for different claims

S_t ▷ Score Threshold

Smart-contract EVALUATEREQUEST

$\Phi_i = [\phi_i^1, \phi_i^2 \dots \phi_i^n] \leftarrow$ GETMERIT$(Ledger, UserId)$

$S_i = \sum_{j=1}^{|C|} \left(w_j * \phi_i^j\right)$

if $S_i \geq S_t \wedge Pool \geq Request$ **then**

return *accepted*

else

return *denied*

end if

end Smart-contract

Experimental Results

Full Compliance - Canon's weights self-organisation

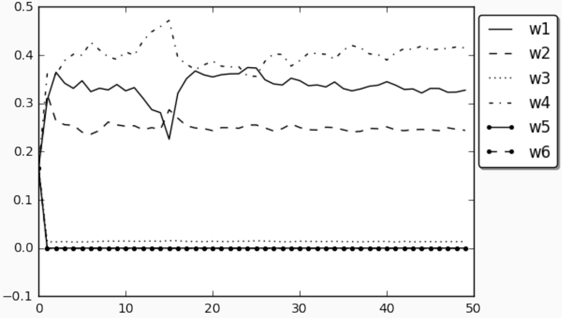


Figure 3: Weights progression in full compliance case.

Relevant canons are selected, so that weights that measure merit gain more importance, while weights where producers have same values go to zero.

Full Compliance - Resource Distribution Results

Table 1: Average users values, at the end of execution.

Physical Facts	
Demand	4096.49 ± 14.24
Accrued	2730.12 ± 18.00
Generated	2730.17 ± 17.85
Allocated	2730.12 ± 18.00
Analytical Facts	
Utility	1361.14 ± 71.96
Satisfaction	0.6493 ± 0.0881
Resources/Need	0.6657 ± 0.0050
Gini Index R/N	0.0038

CPR Manager returns every producer's investment equally, acting as a scheduler, distributing resources to agents efficiently

Full Compliance - Resource Distribution Results (cont.)

Table 2: CPR Manager results after execution for compliant population.

In	81905.27	Rejected	61288
Out	81903.80	Rej. Rule	52003
Out/In	99.99%	Rej. Short	9285
Requests	163583		

System can solve the trade-off between storing resources for urgent future requests and attending immediate demands.

Mixed Population - Canon's weights self-organisation

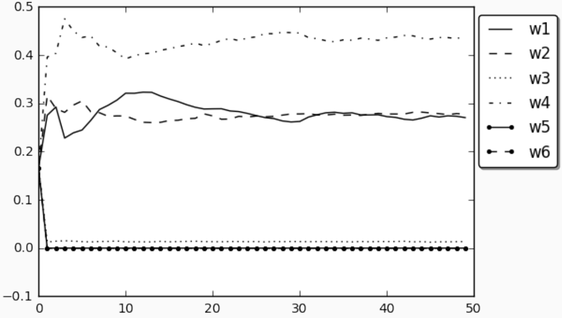


Figure 4: Weights progression without full compliance.

Agents are able to self-organise the relevance of weights in order to benefit the majority of t Weights adapt in favour of the compliant users, as weight 4 directly penalises agents that provide less resources.

Table 3: Average producers' data, at the end of execution, for compliant and non-compliant users.

	Compliant	Non Compliant
Physical Facts		
Need	4090.10 ± 23.12	4096.40 ± 8.87
Resources	3864.74 ± 48.35	454.72 ± 14.71
Generated	2726.43 ± 15.67	2732.29 ± 18.09
Allocated	3838.11 ± 47.41	111.71 ± 14.65
Withheld	26.63 ± 3.40	343.01 ± 7.01
Analytical Facts		
Utility	6963.64 ± 226.47	-10033.87 ± 89.85
Satisfaction	0.98 ± 0.02	0.00 ± 0.00
Resources/Need	0.945 ± 0.0119	0.110 ± 0.0047

Compliant producers are prioritised and have increased satisfaction and utility.

Full Compliance - Resource Distribution Results (cont.)

Table 4: CPR Manager results after execution with mixed population.

In	77888.95	Rejected	59642
Out	77882.77	Rej. Rule	53962
Out/In	99.99%	Rej. Short	5680
Requests	163564		

Decrease in number of rejections shows that smart-CPR can take more precise decision using rules, successfully answering those with legitimate claims.

Conclusion

Final Remarks

- With the automation brought with smart-contracts and blockchain technologies, combined with the computational justice framework, it is possible to solve the problem of distributed supply and demand in efficient and self-organised ways.
- Smart-CPR can efficiently act scheduling allocations of resources in cases of full compliance and prevent abuses of malicious agents.
- Future steps:
    - Test different combinations of agent behaviour;
    - Higher level of decision making - meta-meta arenas
    - Adaptative producer behaviour

Acknowledgemnts

- National Council for Scientific and Technological Development (CNPq), Brazil;
- Diverse colaborators.

Thank you!