The 24th CIRP Conference on Life Cycle Engineering

Operation Mode Study in Cloud Manufacturing Ecosystem

Shengkai Chenb, Shuiliang Fanga,b,∗, Tao Pengb, Renzhong Tangb

aThe State Key Laboratory of Fluid Power Transmission and Control, College of Mechanical Engineering, Zhejiang University, Hangzhou, 310027, China

bKey Laboratory of Advanced Manufacturing Technology of Zhejiang Province, College of Mechanical Engineering, Zhejiang University, Hangzhou , 310027,

China

\* Corresponding author. Tel.: +86-135-8880-3980; *E-mail address:* me\_fangsl@zju.edu.cn

Abstract

With cloud manufacturing, platform operator is able to manage distributed massive manufacturing resources for manufacturing business. Ecosystem of cloud manufacturing has more complicated relationship among individuals than that in existing manufacturing system, every individual makes decisions depend on enriched information, which will help optimize for the overall industry. In this paper, an original operation mode with three extensions are proposed to describe the life cycle vicissitude of each resource. We designed an agent-based model to simulate the ecosystem from the very beginning, and the validation experiment result shows platform have: 1) shorter job queue length and lower resource idle rate with incubation mode; 2) a little shorter job queue length and fewer amount of registered resource with outsourcing mode; 3) the fewest amount of registered resource but a little higher resource idle rate with metabolism mode.

© 2017 The Authors. Published by Elsevier B.V.

Peer-review under responsibility of the scientific committee of the 24th CIRP Conference on Life Cycle Engineering.

*Keywords:* Cloud manufacturing ecosystem; decision-making; operation mode; ecosystem evolution; agent-based simulation

1. Introduction

The relationship among entities in a cloud manufacturing system become more complicated than that in existing manufacturing systems, since the integration of advanced technologies makes it possible for individual to make decisions depend on enriched information. Demanders prefer high quality of performance, while providers prefer high use rate of resource. A cloud platform operator prefers a suitable amount of registered resources to satisfy the needs of the market and for an easier life cycle management of resources. These preferences stimulate the emergence of good service pattern that will lead to the emergence operation mode in the cloud manufacturing ecosystem. Hence, it's important to identify a suitable operation mode to meet most of individuals' preference and to optimize the resource management.

In this paper, an original operation mode is designed to describe the basic decision-makings in cloud manufacturing ecosystem, then three extensions, namely metabolism mode, incubation mode and outsourcing mode are proposed. Finally, an experiment to validate these synthetic operation modes are designed using an agent-based simulation method.

1. Review on cloud manufacturing and simulation

In cloud manufacturing context, platform operator can manage manufacturing service that encapsulated with distributed manufacturing resources intensively with appropriate business model [1], modular approaches and multi-layer architectures are the most common approaches to build a cloud manufacturing platform or system framework [2, 3], Lv used the list of views to depict this multi-layer architecture [4].

Servitization is the key philosophy to operate cloud manufacturing [5]. A service can be created statically which comes along with a provider [2], or can be created dynamically according to task pattern, such method as ‘Multi-Composition for Each Task’ [6] that combines incompetent service as a whole. A service can also be created by AI planning-based automatic composition framework [7].

Simulation approach has been widely used in manufacturing systems on operation planning and scheduling, real-time control, operating policies, performance analysis [8]. In operating policies field, scheduling policies can be tested with simulation performance under given machine conditions [9], machine segmentation policies can be simulated in a combined MRP and Kanban production system [10]. Mourtzis et al. [11] explored a series of simulation-based solutions in industrial practices and concluded that research trends are in Internet- and cloud-based situations.

1. Design of the ecosystem

Before introducing the design of the ecosystem, we specify some basic definitions:

* Provider: the individual who provide resource;
* Resource: the basic task process object with renewable capacity and unique type;
* Demander: the individual who publishes order that contain bunch of tasks;
* Order: the task bundle like a project;
* Task: the basic object need to be processed with resource-type cooperation;
* Task-part: virtual resource-type segmentation unit of one task as squares in Fig. 5;
* Product: the perform result of task;
* Service-call: the basic object that needs to be processed with both resource-type and resource-capacity cooperation;
* Job: the generalization of task and service-call;
* Service: the perform result of a service-call, a group of tasks;
* Machine: the generalization of resource and service;
* Platform: the place where individual interact with other and environment;
* Resource-type cooperation: resource in specific type to process a job simultaneously;
* Resource-capacity cooperation: part of resources in same type to process a service-call simultaneously as in Fig. 3.

Platform is the cradle of the ecosystem and the incubator for manufacturing service, with the cloud manufacturing platform operating, demanders publish orders while providers provide resources, then they make decisions to arrange the resources to perform tasks that decomposed from order. Most recent researchers e.g. Wu et al. [12], describe this operation procedure in cloud manufacturing as a tri-group user model that contains: 1) users/customers, 2) application providers and 3) physical resource providers. Inspired by this model, we design the original operation mode shown in Fig. 2 as the basis. In this mode, individual executes activities that interact with others depicted by object flow (full lines) and information flow (dashed lines). A single order can be described by an activity-on-node (AON) network where the node represent the task and the arc the precedence relation. Each task needs to be performed with resource-type cooperation as listed in Tab. 1, and the selected resource cannot start to process the task until all of other selected resources are ready, what each resource actually processes is the task-part and we call the task-part is **active** when being processed, **semi-active** when selected resource is waiting as shadow in Fig. 5, **inactive** when this part is just assigned to the job queue. Product, the performance result after the processing and assembly procedure, will be delivered to demander, then demander change the rank value of the selected resource owners according to the review of the product.

* 1. Nomenclature and assumptions

**Nomenclature**

Order that come with demander, who can be inquired by

Task belongs to

Process duration of

Expect quality of product after the finish of

Release time of all

Actual finish time of

The set of predecessor of , determined by order and some assign procedure

Resource that come with provider, who can be inquired by

Task-part quality produced via resource

Capacity of at time

Available capacity of at time

The list of inactive job queue of at time with sequence

The list of semi-active job of at time

The set of active job of at time

Ideal finish time of in for schedule at

Remaining process time of in for schedule at

Resource type subset required by

Required amount of resource with type by ,

Service-call generated by provider

Process duration of

Release time of

The set of predecessor of

Resource type subset required by or provided by

Service that generated after the finish of

Product quality produced via service

Resource member set of

Resource member’s capacity contribution set of

Need resource capacity of with type

Subset of that consists of all the capacities of type resources in

The list of job queue of at with sequence

The set of active job of at

Resource candidates set for to select

Resource candidate types set for

Service candidates set for to select

Resource candidates set for to select

Resource candidate types set for

Rank inquire function about provider

Owner inquire function about resource or service

Type inquire function about resource type

Bold font of the variable () means the temp set of a bunch of these variables

Since demander and provider continuous arrives, there is no upper bound of the subscripts (). To scope our research, we make some assumptions as follows for the original mode, while in other extension, some of the assumptions will be modified.

* Operate of the ecosystem starts from the very beginning that no demander or provider is registered;
* Each single task should be assembled by its task-parts, and these parts should be processed simultaneously;
* The quality of product is determined by the worst quality of the selected resource;
* Resource are renewable that the available capacity will be return to when the process procedure finished;
* Resource quality comply normal distribution with given mean and standard deviation by provider when registered;
* Cooperate and process procedure cannot be interrupted;
* Provider can only schedule task-parts in inactive status;
  1. Master plan for original and extended modes

In original mode operating of ecosystem begins with void, then there comes the registration of provider and demander. A single **order** consists of a set of tasks, the tasks are interrelated by kinds of constraints. First, precedence constraints force **task** not to be started before all its immediate predecessors . Second, performing the tasks requires resources with limited capacities. Third, resources cooperation requires all the task-parts in active status. A single **resource** () belongs to only one type. While being processed, task requires units capacity of the resources with each type during every period of its non-preemptable duration . Each resource has a limited capacity and available capacity at any point in time . This plan is much like the settings in RCPSP[13] except that the task here need to be processed with resource cooperation.

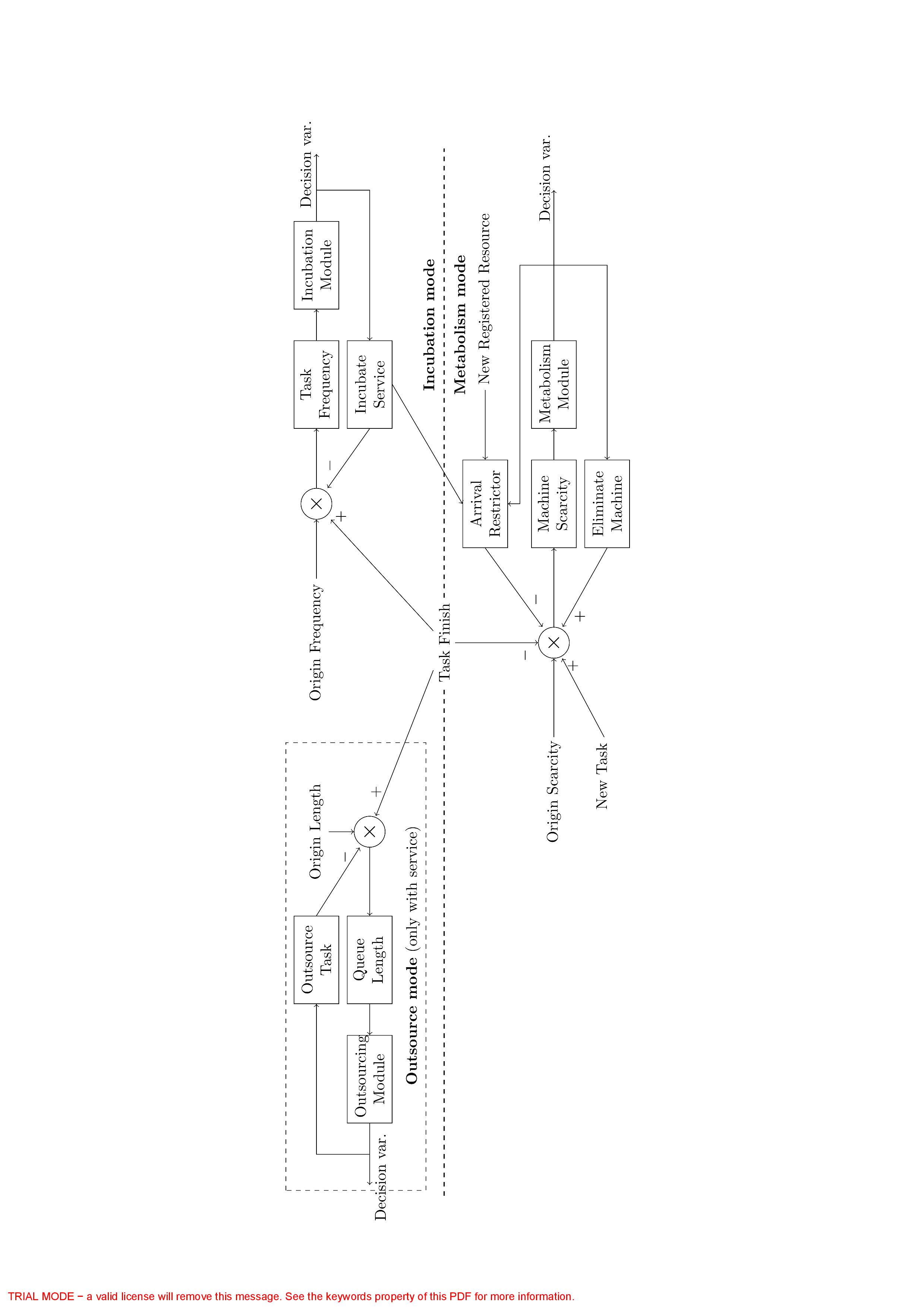


Fig. 1. Three Extended modes

The life cycle management of resource contains its vicissitude, which is the design basis of proposed extended modes Fig. 1. Incubation and outsourcing mode are options for provider, while metabolism mode is option for platform operator, and the outsourcing mode can only be applied if incubation mode is applied because only the task that processed by service can be outsourced.

* 1. Original mode

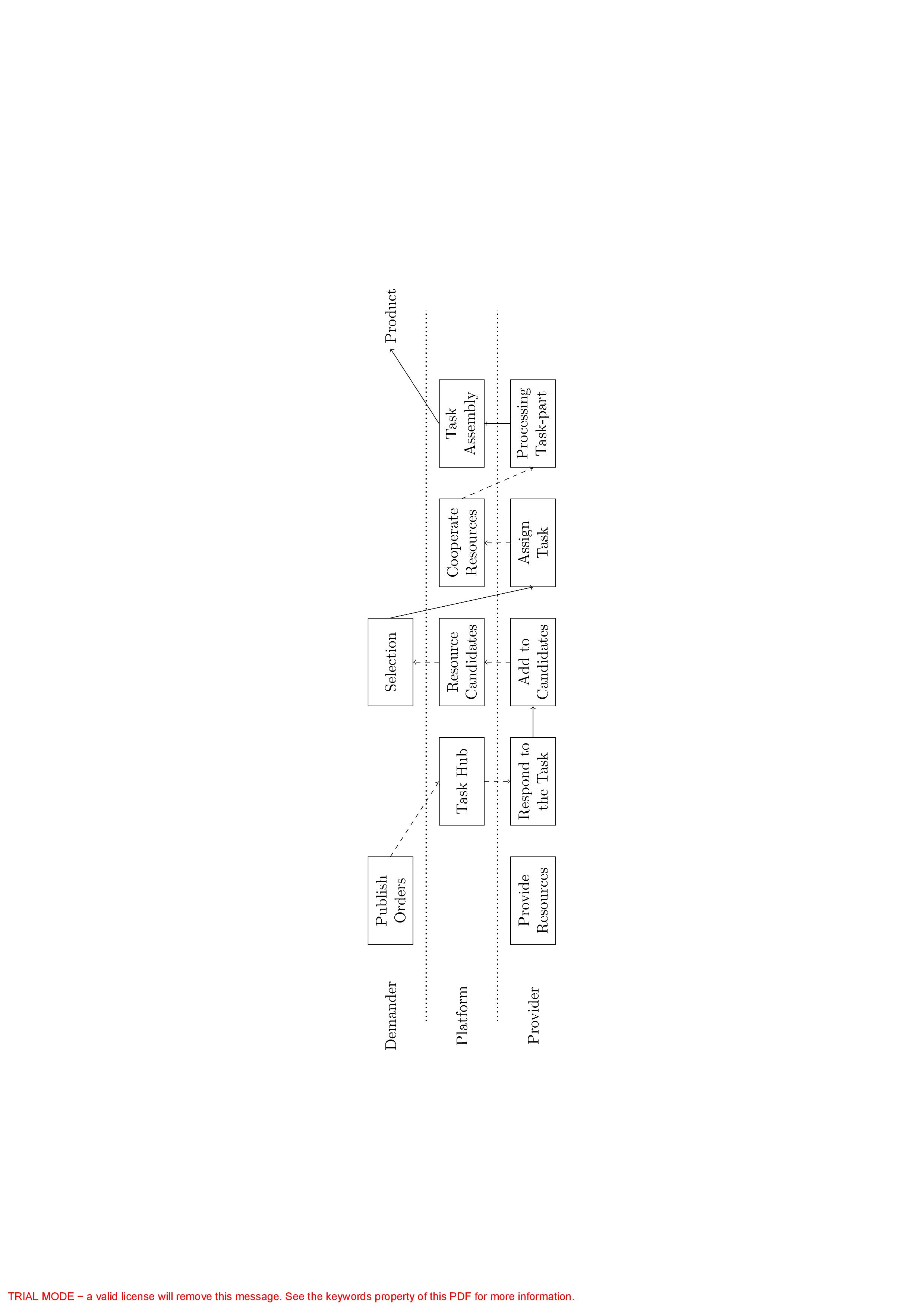


Fig. 2. Original mode

A provider makes decision at time to respond to task if the belonging resource was type-matched with (). If , the provider will respond to the task need, demander of will add to and to . If is finally selected, available capacity of will change to . This available capacity will be restored after the finish of part.

A demander makes decision about resource selection from when . Without loss of generality, we suppose at time , the decision-making of demander can be described as follows,

 (1)

  (2)

  (3)

 (4)

*s.t.*

 (5)

  (6)

The multi-objective in Eq. 1 aims at high quality, high resource owner rank and low waiting queue length. Eq. 2 ensures the resource capacity is capable to process the task-part, Eq. 3–4 restrict the resource candidates’ type in the task configuration, Eq. 5–6 describe decision variable, the demander should make decision to select resource in each type.

If was selected by the demander of , then the provider will add part to if , , or to otherwise. If all the part of are in semi-active status, then these providers will change all the task-part status from semi-active to active and add to .

* 1. Incubation mode

Incubation mode is one extension for the original mode, the purpose of this mode is to remove the cooperation and assembly procedure in advance by gathering types of resources with certain quota into manufacturing service, which will be incubated as shown in incubation part of Fig. 1. After the finish of task-part, provider record the resource configuration as task frequency, when the frequency value reached some point, provider will decide to incubate such a task-specified service for the future performance. As the example in Fig. 3, if task in Tab. 1 was finished, the first thing of incubation for the resource provider is to publish a job named **service-call** (), which is similar to task except for the capacity dominance feature , which means capacity of selected resource will not be restored after the processing.

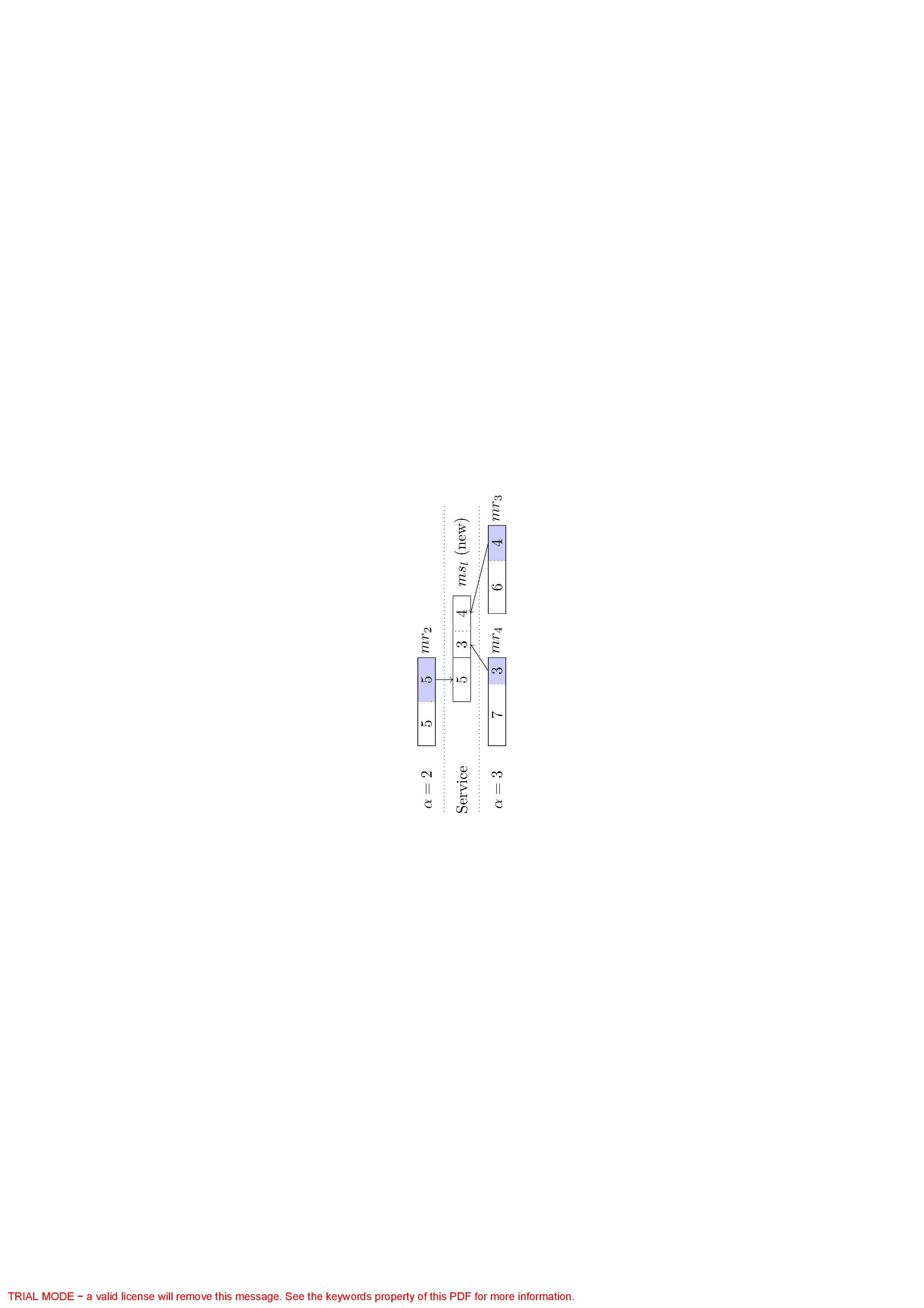


Fig. 3. Service incubation from in Tab. 1

The process result of one service-call is manufacturing **service** as shown in Fig. 3, which is actually a set of resources that may come from selected providers in the system, it’s possible for more than one resources in a same type to make contributions to the incubation of one service. No more cooperation and assembly procedure are one pro of service, and product quality will no longer be restricted to the worst quality of resources for the complementary effect, while one con of service include that service can only perform specified task. Now we can generalize task and service-call into **job**, resource and service into **machine** for the follow discuss.

Apart from response for as we explained we explained above, there are 2 other cases in incubation mode:

1. response for

If was type-matched with (), as long as , provider of will respond to the service-call. If was finally selected, both available capacity and capacity will change as Eq. 7a–7b, and these capacity will not be restored even after the finish of part. Provider of will add to and to .

2. response for

Service is task-oriented machine, so if type-matched (, ), provider of it will respond to as soon as possible, demander will add to .

Apart from demander of select that , , there are 3 other cases in incubation mode:

1. Select for when

Provider of select resource from is similar to Eq. 1–6, but the difference is that the here is also a set of selected resources with type , is subset of decision variable . Selected resources capacity will change as Eq. 7a–7b, and if the sum of capacity quantity in less than the required, this selection will not to be executed.

 If  (7a)

 otherwise (7b)

2. Select for when

This situation implies that the resource candidates are not enough and there exists service candidates.

 (8)

 (9)

 (10)

*s.t.*

 (11)

 (12)

Similar to Eq. 1, Eq. 8 is also a multi-objective function that aims at high quality, high service owner rank and low job queue length. Eq. 9–11 explain the service quality distribution parameters, Eq. 12 is the decision variable to select one service in .

3. Select machine for when

In this situation, demander of will select ether a bunch of resources with cooperation or one single service, so the key idea here is to pre-select optimal resources and optimal service, then to compare these two optimal options to select the better one.

 (13)

 (14)

 (15)

 (16)

 (17)

 (18)

 (19)

*s.t.*

 (20)

 (21)

 (22)

 (23)

 (24)

 (25)

Similar to other situations, Eq. 13 is a multi-objective function that aims at high quality, high rank and low waiting queue length. Eq. 14 and Eq. 22 are the optimal decision in independent conditions, Eq. 15, Eq. 16 and Eq. 17 are the virtual rank value, virtual queue length and virtual quality value that are set in the worst cases. Eq. 17–21 are the virtual quality value calculate procedure. Eq. 25 is the decision to choose one of these two partial optimal decision.

Apart from assign , there are 2 other cases in incubation mode:

1. Assign to

If is selected by provider of at time , the assign condition is more restrict, it should be changed into , for the capacity dominance feature of service-call, we need to assign this type of job one by one. Then, the provider of should change the predecessor set:

 (26)

and all the task assign after , we say will set:

 (27)

And will be changed with the same amount in Eq. 7a–7b.

2. Assign to

Assign to is very simple and there will be no semi-active status for , hence it will add to if , or to otherwise.

* 1. Outsourcing mode

Outsourcing mode is one extension for the original mode when incubation mode is applied. As outsourcing part shown in Fig. 1, the idea of outsourcing for service is to republish active or inactive task to platform again i n order to reduce its job queue length to enhance the probability to be selected by new tasks. For each single task , the only condition to make the outsource decision is that if the maximum delay () of task in both status decreases. This decision highly depends on the estimation of other resources and services’ performance status. Outsourcing mode makes it possible to paralleled process one job.

* 1. Metabolism mode

Metabolism mode is one extension for the original mode to control the number of individual in the system by both restrict the arrival and eliminate the current members. As metabolism part shown in Fig. 1, we can define machine scarcity as .

Platform restrict the arrival of new machine by block the registration, eliminate the current machine by their quality and owner’s rank.

* 1. Provider schedule the jobs in machine

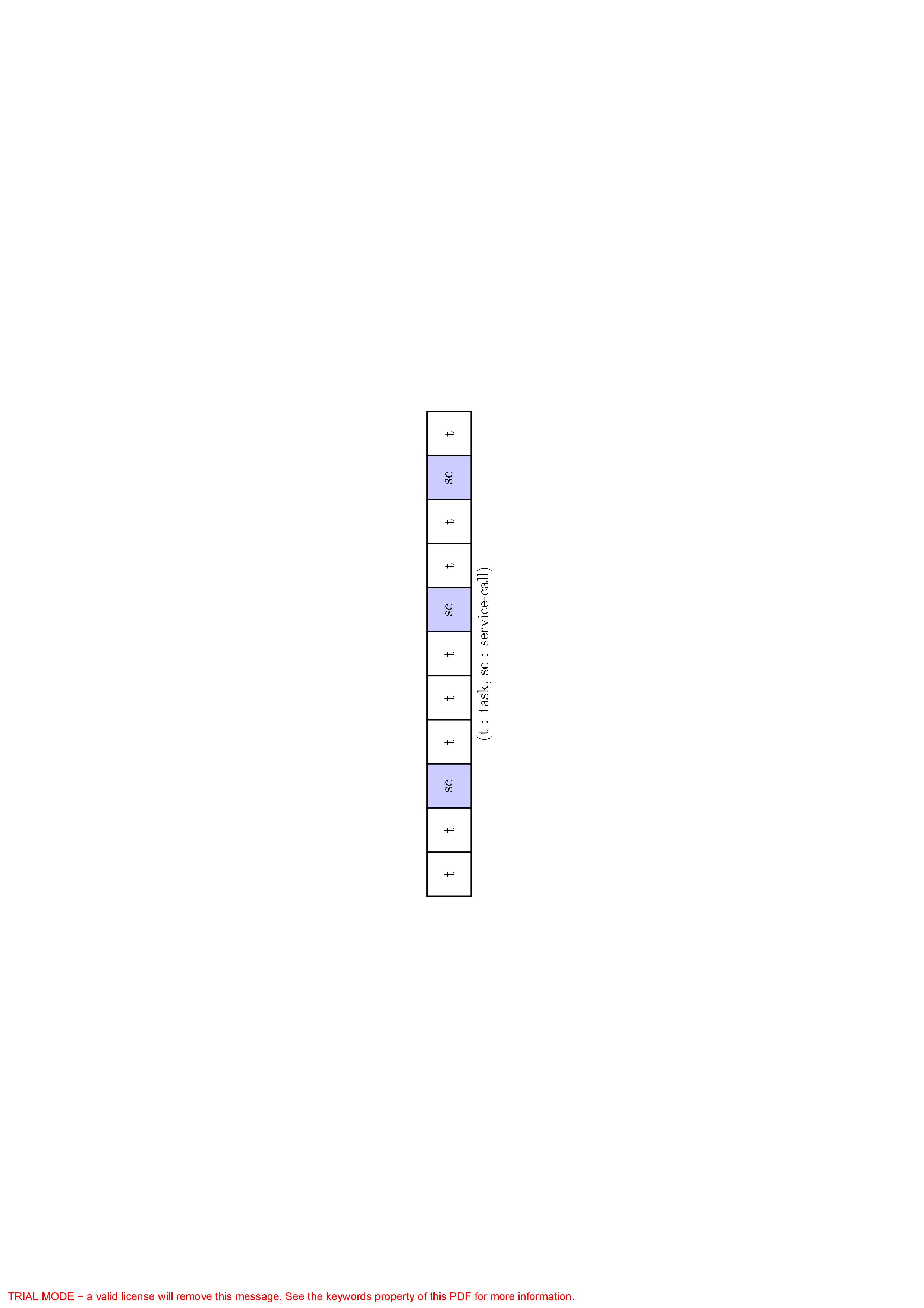


Fig. 4. Simple illustration for one

In order to reduce the idle rate, provider will schedule the inactive jobs on their machines. Since the schedule in service is a single machine scheduling problem that not on what we focus in this study, we only discuss the job scheduling in resource. Since the resource capacity dominance feature of service-call, it’s necessary to suspend service-call in inactive status until all the job before it in are finished, at the meantime, all the job after service-call should be stay in inactive status until the service-call is finished, as shown in Fig. 4, service-call likes partition plate in . Therefore, provider can only makes decision on the schedule of tasks before the first service-call in when one of the active job in was finished at , we denote the set of schedule task at this time as .

Specifically, a simple instance with configuration Tab. 1 and schedule chart Fig. 5 will elucidate the settings.



Fig. 5. Simple instance schedule chart with 4 resources in 3 different types

Since the task may come from different orders, we here use the single uniform denotation  to distinguish these tasks and their related variables. In this instance, as shown in Fig. 4, horizontal dotted line constrained the available capacity of the resource for the subsequent jobs. Each finish of service-call will make the horizontal dotted line lower and it will never get higher again unless the related service is repealed.

Table 1. Simple job configuration

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Job () | Entity class |  |  |  |  |  |
|  | Task | 0 | 5 | 7 | 0 | 7 |
|  | Task | 7 | 5 | 5 | 4 | 5 |
|  | Task | 6 | 0 | 5 | 2 | 3 |
|  | Task | 9 | 4 | 6 | 1 | 2 |
|  | Task | 1 | 0 | 3 | 2 | 3 |
|  | Service-call | 1 | 0 | 3 | 0 | 1 |
|  | Service-call | 0 | 5 | 7 | 7 | 1 |

In every single resource, a schedule is given by a vector of ideal finish times , . Even though provider will not schedule the finish time of these job cannot be determined because the waiting in the cooperation. We ignore the type subscript because all the job assigned here are already type-matched. The schedule model is:

 (28)

 (29)

  (30)

 (31)

*s.t.*

  (32)

  (33)

  (34)

The schedule aim for each resource Eq. 28 is to minimum the maximum delay of jobs. Eq.30 ensures that all predecessors of each job finished before the job itself. Eq. 32 means that the finish time of activate job is determined. Eq. 33 ensures the capacity restriction at every time period and Eq. 34 defines the extreme situation of the finish time. Since Eq. 33 is a time dependent function, the schedule model cannot be solved with mixed integer programming (MIP) techniques.

1. Experiment and result Analysis

Because of the autonomous and self-directed features of individuals in the manufacturing ecosystem, we use ABMS technique [14, 15] to study such a complex system. Repast Simphony [16] package was utilized in this experiment.

* 1. Experiments

We design experiments to repeatedly simulate the operating of ecosystem from the very beginning with mode combinations in Tab. 3 group by group, which are the prototypes of feasible cloud manufacturing operating modes. Every single simulation goes with the main flow as show in Fig. 2. We use RanGen [17] to generate dataset in the well-known Patterson format as the continuously arriving order for the simulation input, related parameter settings are listed in Tab. 2. First 5 parameters are defined in [17], is the parameters of the arrival of user in simulation, is the initial capacity for .

Table 2. Order generating parameters setting

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | OS | CI |  | RF | RC |  |  |  |
| 10 | 0.3 | [2,3] | 12 | 0.3 | 0.5 | 0.2 | 0.3 | [20,30] |

We assume that providers and demanders are arriving as the Poisson process, in order to make sure the coming need resource capacity rate will not exceed the average coming resource capacity rate in average to prevent the task explosion, and we make sure that,

 (35)

Table 3. Experiments mode grouping

|  |  |  |  |
| --- | --- | --- | --- |
|  |  | No metabolism | Metabolism |
| Resource-only |  | Mode 11 | Mode 12 |
| Incubation | No outsourcing | Mode 21 | Mode 22 |
| Outsourcing | Mode 31 | Mode 32 |

* 1. Result and analysis

Experiment in each group is simulated with same random seed to make no difference on the irrelevant configuration to our validation. After bunch of experiments with different random seeds, take random seed 776189616 as example, we run the simulation for 6000 tick time and the experiment’s results are displayed in Fig. 6 and Tab. 4, we find:

1) Most dotted lines are above the full lines with the same series in means that metabolism mode need lower number of provider and resource, with the price of higher number of queue length of the tasks in resources in the system to deal with the same amount of needs. Full lines are not much above dotted lines in means that task finish rate will be a little lower with the metabolism.

2) and both represent the job queue length, triangle and square line turned lower in means that incubation mode can help reduce the waiting of job, these two type of line above the circle line means the incubation mode do reduce the resource idle rate.

3) Triangle lines are a little above the square lines in means that outsourcing mode needs more resources to operate and can only reduce the job queue length a little.

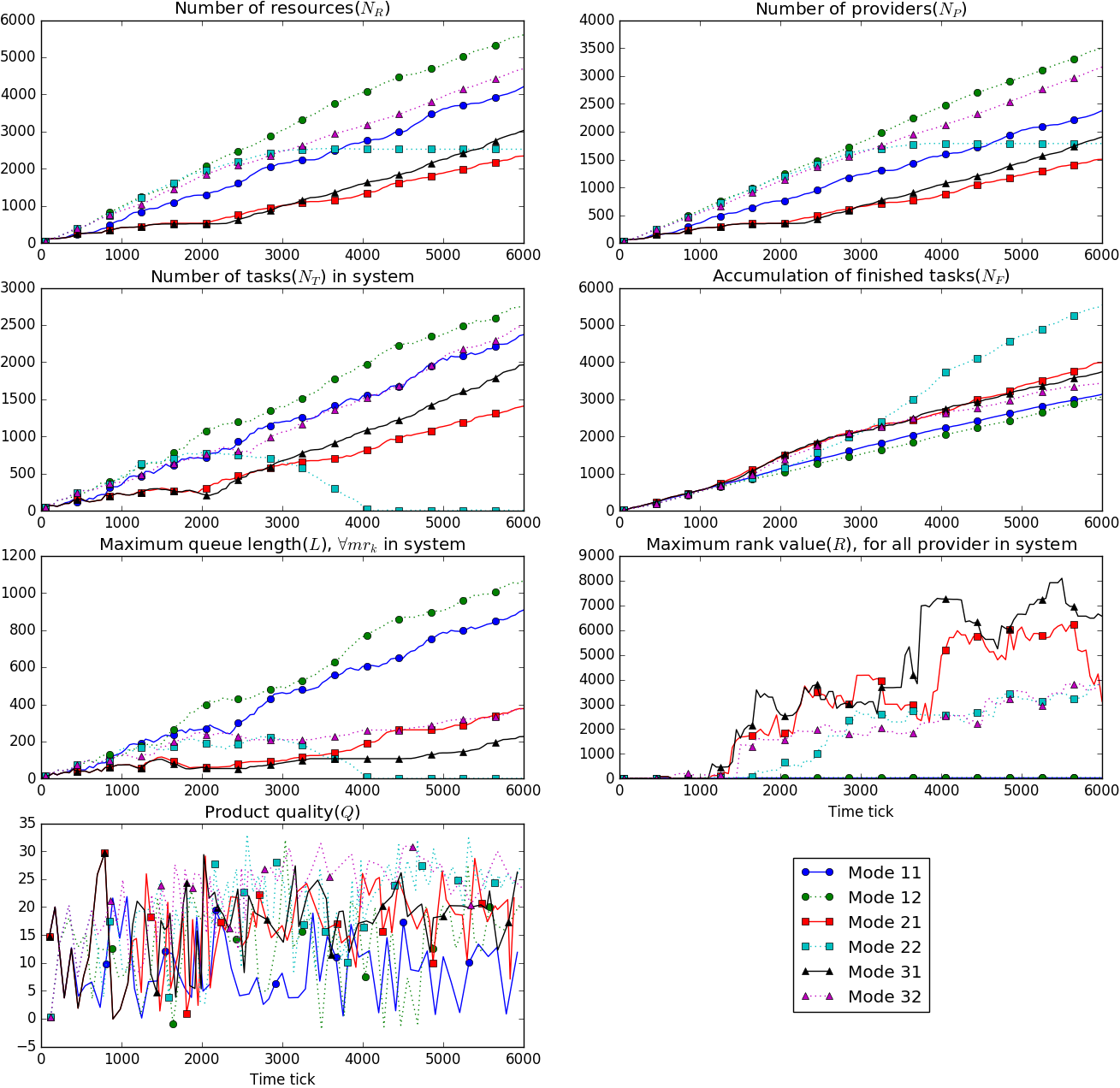


Fig. 6. Observed variable change with time

4) There is no big difference among all the 6 modes in rank change. Provider in metabolism mode will get lower rank value for they cannot stay in the system for a longer time to get higher rank value. Provider in Mode 11 even will not promote their rank value, which means that the metabolism rate is very fast if without incubation mode.

5) Mode 22 is a very special mode that reached the balance within 6000 ticks, that the number of provider and resource changes little with time goes by. Because our experiment setting comply Eq. 37, the coming need capacity rate is lower than the coming capacity rate, Mode 12,22,32 will finally reach balance in the long run theoretically, but the appearance of service-call and metabolism mode itself is full of uncertainty, thus it’s hard to predict when other metabolism related mode will reach the balance

Table 4. Average observed values

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Mode |  |  |  |  |  |
| Mode 11 | 1159.36 | 1644.201 | 1177.601 | 2053.727 | 8.388 |
| Mode 21 | *636.769* | *2031.215* | **692.716** | **1066.947** | 16.016 |
| Mode 31 | 779.413 | 1996.021 | *782.272* | *1203.748* | 15.174 |
| Mode 12 | 1439.831 | 1522.923 | 1797.883 | 2954.113 | 14.057 |
| Mode 22 | **343.444** | **2464.407** | 1316.952 | 1957.226 | *20.251* |
| Mode 32 | 1164.409 | 1907.011 | 1597.742 | 2428.155 | **20.652** |

6) With the average data value shown in Tab. 4, metabolism with incubation but not with outsourcing mode is one ideal combine mode to maintain the system, resource will self-configured and controlled to justly meet the task need.

1. Conclusions, Limitations, and Future Research

In the extended modes we designed for cloud manufacturing ecosystem, incubation mode can realize a feasible solution for the formation of manufacturing service, shorten the job queue length and reduce the resource idle rate; outsourcing mode can cut down the amount of registered resource; metabolism mode can also cut down the amount of registered resource in price of a little higher resource idle rate. The combine of incubation and metabolism mode makes the maintenance of platform much easier. However, we only proposed 3 extensions to combine and it may not fully describe the operation mode of cloud manufacturing system, it may also limit the evolution direction of the ecosystem, hence we will design more extensions. The assignment of service-call is oversimplified to prevent complex consequence, we will design new approaches to assign this job.

Acknowledgements

This work is supported by the National Natural Science Foundation of China (No. 71571161), the National High Technology Research and Development Program of China (863 Program) (No. 2015AA042101), and the Fund of Key Lab of AMT of Zhejiang Province (No. 2015QN01).

References

[1] X. Xu, From cloud computing to cloud manufacturing, Robotics andComputer-Integrated Manufacturing 28 (1) (2012) 75 – 86.

[2] F. Tao, L. Zhang, K. Lu, D. Zhao, Research on manufacturing grid resource service optimal-selection and composition framework, Enterprise Information Systems 6 (2) (2012) 237–264.

[3] O. F. Valilai, M. Houshmand, A collaborative and integrated platform to support distributed manufacturing system using a service-oriented approach based on cloud computing paradigm, Robotics and Computer-Integrated Manufacturing 29 (1) (2013) 110–127.

[4] B. Lv, A multi-view model study for the architecture of cloud manufacturing, in: Digital Manufacturing and Automation (ICDMA), 2012 Third International Conference on, July 31 2012-Aug. 2 2012, pp. 93–97.

[5] B.-H.Li, L.Zhang, S.-L.Wang, F.Tao, J.Cao, X.Jiang, X.Song, X.Chai, Cloud manufacturing: a new service-oriented networked manufacturing model, Computer integrated manufacturing systems 16 (1) (2010) 1–7.

[6] W. Liu, B. Liu, D. Sun, Y. Li, G. Ma, Study on multi-task oriented services composition and optimisation with the 'Multi-Composition for Each Task' pattern in cloud manufacturing systems, International Journal of Computer Integrated Manufacturing 26 (8) (2013)

[7] S. C. Oh, D. Lee, S. R. T. Kumara, Effective web service composition in diverse and large-scale service networks, IEEE Transactions on Services Computing 1 (1) (Jan.-March 2008) 15–32.

[8] J. S. Smith, Survey on the use of simulation for manufacturing system design and operation, Journal of Manufacturing Systems 22 (2) (2003) 157 – 171.

[9] I. Sabuncuoglu, O. B. Kizilisik, Reactive scheduling in a dynamic and stochastic fms environment, International Journal of Production Research 41 (17) (2003) 4211–4231.

[10] T. Felberbauer, K. Altendorfer, A. HÃijbl, Using a scalable simulation model to evaluate the performance of production system segmentation in a combined mrp and kanban system, in: Proceedings of the 2012 Winter Simulation Conference (WSC), 9-12 Dec. 2012, pp. 1–12.

[11] D. Mourtzis, N. Papakostas, D. Mavrikios, S. Makris, K. Alexopoulos, The role of simulation in digital manufacturing: applications and outlook, International Journal of Computer Integrated Manufacturing 28 (1) (2015) 3–24.

[12] D. Wu, M. J. Greer, D. W. Rosen, D. Schaefer, Cloud manufacturing: Strategic vision and state-of-the-art, Journal of Manufacturing Systems 32 (4) (2013) 564–579.

[13] R. Kolisch, S. Hartmann, Heuristic algorithms for the resource constrained project scheduling problem: Classification and computational analysis, in: J. WÄ ´ Zglarz (Ed.), International Series in Operations Research & Management Science, Vol. 14, Springer US, 1999, pp. 147–178.

[14] C.M.Macal, M.J.North, Agent-basedmodelingandsimulation, in: Winter Simulation Conference, Winter Simulation Conference, Austin, Texas, 2009, pp. 86–98.

[15] M. J. North, C. M. Macal, Managing business complexity: discovering strategic solutions with agent-based modeling and simulation, Oxford University Press, 2007.

[16] M. J. North, N. T. Collier, J. Ozik, E. R. Tatara, C. M. Macal, M. Bragen, P. Sydelko, Complex adaptive systems modeling with repast simphony, Complex Adaptive Systems Modeling 1 (1) (2013) 1–26.

[17] E. Demeulemeester, M. Vanhoucke, W. Herroelen, Rangen: A random network generator for activity-on-the-node networks, Journal of Scheduling 6 (1) (2003) 17–38.