Creation of Robust Schedule for Profit Based Cooperation

Jiří Hodík

▶ **To cite this version:**

Jiří Hodík. Creation of Robust Schedule for Profit Based Cooperation. 9th IFIP WG 5.5 International Conference on Balanced Automation Systems for Future Manufacturing Networks (BASYS), Jul 2010, Valencia, Spain. pp.160-167, 10.1007/978-3-642-14341-0_19 . hal-01060720

HAL Id: hal-01060720
https://inria.hal.science/hal-01060720

Submitted on 16 Nov 2017

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Distributed under a Creative Commons Attribution 4.0 International License

Creation of Robust Schedule for Profit Based Cooperation

Jiří Hodík

Gerstner Laboratory – Agent Technology Center, Faculty of Electrical Engineering,
Czech Technical University in Prague, Technická 2, Praha 6, Czech Republic
hodik@labe.felk.cvut.cz

Abstract. Method for robust schedule preparation for domain of profit based cooperation is introduced in this paper. The method aims schedules, which are prepared and concluded step-by-step during the team formation phase. The method takes into account domain specifics, like incomplete information about the partners, and their various reputation and skill levels. The method is based on critical path dispatching rule, which is widely used in heuristics for *np* hard and continuous scheduling problems. Experimental results are presented for various workflow structures and other configurations of the domain.

1 Introduction

In the domain of profit based cooperation, the cooperators are motivated to join (and to stay in) the team (e.g. Supply Chain and some types of Virtual Organizations) by expected profit. We concentrate to teams with one leader subcontracting the other partners mutually independently. Sometimes the leader has to conclude contracts with the others without having prepared whole schedule; it is to be prepared online during the negotiation about involvement the partners. Contrary to the intra-organizational scheduling, this scheduling adds requirements and constraints to scheduling methodologies. Together with complexity of the scheduling problem it is too hard to prepare valuable estimation of schedule until it is negotiated with the partners. Moreover, the leader has limited access to information about the partners' resources and such information may be distorted either by intention of partners to improve impression about their competencies, or by inexperienced guess of them.

When any already concluded due-date is not kept and if there is not enough slack, the processing of consequent tasks is delayed. It is critical mainly when it is not possible to reschedule the consequent tasks either because of high utilization of facility, or because of high penalties for not keeping the concluded due-dates.

The targeted domain relates to the supply chain planning and scheduling models class, which includes medium planning models mainly on an entire network level (e.g., project planning and scheduling or lot sizing models), as well as detailed scheduling models mainly on work-centers levels (e.g., job shop scheduling). The features crucial for the top level schedule creation are:

- *Members.* They may be autonomous, self-oriented and distributed.
- *Limited access to information.* Nobody has direct access to the partners' internal information, neither the team leader.
- *Partners with limited reliability.* Presumption about partners' trustworthiness may be helpful in negotiation about resources allocation.
- *Concluded contracts.* Withdrawing from a contract or changing it is limited by the concluded rules and the affected partner's willingness.

There are several scenarios for team creation phase; all of them provide the plan and schedule for further team operations. The schedule is pre-negotiated with the partners before its conclusion, or involvements of the partners are being concluded continuously during the scheduling process. Reasons for continuous concluding are: (i) partners provide an expected completion time of the task but they do not share knowledge about available resources; and (ii) partners' resources may be shared within the partner; once dedicated for a task they cannot be used for the other one.

The scheduling algorithm is interlinked with a protocol for negotiation about tasks processing commitments. Often, it is possible to pre-negotiate involvement of the members, or mechanisms for its update according to current state of the team exist. There are several works extending basic negotiation protocols by non-bounding offers and commitments to improve quality of the consortium and its schedule like LAP [5], C-CNP [7], RC-CNP [1], and RBVO formation protocol [8]. Although any such mechanism for involvement modification may be implemented, there is a condition of available resources. Without resources, the schedule update is excluded although the negotiation protocols would support the updating process.

2 Scheduling Methods

Many scheduling methods for (not only hard) scheduling problems are based on priority rules comparing tasks priorities from attributes of tasks, machines, and actual state of the world. The rules are *static*, or *dynamic* (time is important as well). There are several schedule measures and objectives. Example of a schedule objective is the *makespan*, which is defined as the latest completion times of mutually related tasks.

2.1 Critical Path Method

The Critical Path Method (CPM) is a common method for scheduling set of tasks with defined precedence constraints. The objective to be optimized is the makespan. The method concentrates to finding and optimization of *critical path* under these constraints: (i) the processing time of a task is fixed, (ii) no other resource than machine time is required, and (iii) there are enough of parallel machines. The CPM inspired for Critical Path dispatching rule, which employs precedence constraints and processing times of tasks. From them the strings of processing times are counted and the head of the longest one is given the highest priority.

2.2 Robustness of Schedule

The robust schedule is required when any rescheduling may negatively influence the schedule objectives (e.g., too expensive, late, or no alternate resources). The robust schedule has *"the ability to satisfy performance requirements predictably in an uncertain environment"* (Pape in [4] in [2] by Davenport et al.). Pinedo presents four examples how to create a more robust schedule [6]:

- *Inserting idle/slack times.* Depending on a rule, the idle times may be equal or depending on size/priority of foregoing task.
- *Scheduling less flexible jobs/tasks first.* Each task may be influenced by aggregated perturbations of its predecessors, for the less flexible ones this risk should be minimized.
- *Postponing the processing of an operation only if really needed.* Contrary to the inventory cost point of view it is better to start as soon as possible.
- *Bottleneck machine utilization optimization.* As the bottleneck has high impact on
- facility capacity and makespan, is should not wait for a task.

We suggest extending the Pinedo's list of potentials to increase the robustness by:

- *Reliable partners.* The ones with higher probability to keep concluded dates should be preferred.

Various method for computation of idle times is explored e.g. by Davenport et al., who concentrate to using robustness technique for scheduling of job-shops with totally ordered tasks in domain with full access to control over the machines [2].

To evaluate robustness of the schedule, the measure must be defined. The robustness may be counted from the schedule as it is prepared, or evaluated after the scheduled tasks (real or simulated) execution. Pinedo defines a concept, which evaluates robustness of the original schedule. The measure is based on the amount of weighted slacks between the tasks' completion times and due-dates [6]. Kouvelis and Yu suggest criteria to evaluated schedule robustness from simulated performance [3]:

- *Absolute robustness.* The worst scenario performance is the key.
- *Robust deviation*. Each scenario performance is evaluated against the optimal one to get the deviation. The worst deviation got is the key.
- *Relative robustness.* Similar to the robust deviation but the percentage deviation from the optimal case for the scenario is used.

3 Scheduling Process

The presented scheduling algorithm is based on CPM. The *makespan* and *schedule robustness* are the optimized objectives. The idea is that the critical tasks should be preferred during scheduling. Contrary, as the expected processing times are known after their negotiation, they may differ from the nominal ones. Therefore the critical

path is not known until the whole schedule is prepared. It is being estimated and modified during the scheduling process. Simultaneously, keeping the negotiated times is not guarantied and therefore there is a request for robust schedules to be able to process whole job of the team.

3.1 Formal Description of the Scheduling Problem

From the task definition a nominal processing time is known. Actually, due to various capabilities and available resources, the processing times negotiated with the machines[1] may differ. While the capabilities are supposed to be constant during the lifecycle of the VO, the resources may vary in time.

The nominal processing time of the task[2] j is p_j. For starting time S_j, the nominal completion time is $C_j = S_j + p_j$. For processing of task j on machine i the concluded completion and processing times are C_{ij} and p_{ij}. There may be also an expected completion time C'_{ij}, which is delayed by δ_{ij}. The expected processing time is $p'_{ij} = C'_{ij} - S_{ij}$; relation $p'_{ij} \geq p_{ij}$ may be expected.

For estimation of the delay, only following is available: the task and its complexity (represented by p_j), concluded processing time p_{ij}, and confidence in machine (given as some reputation model R_i). The expected delay for the task j is $\delta_{ij} = f(p_j, p_{ij}, R_i)$. As the concluded processing time p_{ij} is counted from the machine's available resources, p_j may be omitted and therefore $\delta_{ij} = f(p_{ij}, R_i)$.

3.2 Algorithm Overview

Estimation of the critical path in any scheduling step is based on already concluded tasks processing (on the expected completion times), and nominal processing times (for scheduled yet part of the schedule). The following is defined:

- *JOB* is a list of all tasks, $JOB = \{t_1, \ldots, t_n\}$, where t_j for $1 \leq j \leq n$ is a task from the list, and n is number of tasks in the list. For any couple of tasks the finish-to-start transitive precedence constraint $t_1 \prec t_2$ may be defined. The precedence graph must be acyclic.
- *CLOSE* is a list of already scheduled tasks. For every t_k its starting time, and concluded and (if applied) expected completion times are known.
- *OPEN* is a list of not yet scheduled tasks. Let *OPEN*' denote a subset of tasks from *OPEN* having their precedential tasks scheduled.

---

1 To be consistent with common terminology, the work-centers of the members are called machines.

2 In the definitions *j*, *k* are indexes of tasks, and *h*, *i* indexes of responsible machines.

For $\forall t_k \in OPEN$ the earliest possible starting time is denoted by the latest completion time from set of completion times of all predecessors: $S'_j \geq \max\{C_k : t_k \prec t_j\}$. If $\forall t_k (t_k \prec t_j \rightarrow t_k \in CLOSE)$ then t_j may be scheduled with starting time S'_j, otherwise S'_j denotes that final $S_{ij} \geq S'_j$.

There are various priority rules for selection of the next task to be scheduled (t_{NEXT}) from the $OPEN'$ list. We have adopted following ones:

- *Any ready task rule*, by that any task ready to be scheduled is chosen.
- *Nominal critical path rule*, the next task is chosen according to the critical path counted from nominal processing times.
- *Current critical path rule*, the next task is chosen according to the critical path, which is updated according already existing schedule fragments.
- *Longest chain of successor rules*, the next task is chosen as the head of the longest chain of not scheduled tasks yet.

When the t_{NEXT} is defined, the negotiation about its assignment and completion time is performed. We consider basic negotiation protocol which for defined starting time provides a set of completion times proposed by the interested machines. The decision process for determination of the winning proposal takes into account the proposed completion time and (if applied) reputation model of the proposer. We have adopted two rules for winning offer determination:

- The *basic rule* for offer determination: $C_{\min}(C_{hj})$, where C_{hj} is by machine h proposed completion time for task j.
- The *reputation rule* for offer determination: $C'_{\min}(C'_{hj})$, where C'_{hj} is the expected completion times counted from reputation models of the proposers ($C'_{hj} \geq C_{hj}$).

Next step to increase the schedule robustness is adding the *idle time* to the concluded completion time. The idle time $idle_j$ of the task j postpones earliest time for starting consequent tasks ($S_k - C'_{ij} \geq idle_j; t_j \prec t_k$). In this work, the rules for idle time determination do not consider whether the task is the earlier one or the latter one (such information could be also beneficial as described in [2]). Our rules are:

- No *idle time rule*, $idle_j = 0$.
- The *nominal processing time rule* based on requested robustness and nominal processing time, $idle_j = f(p_j)$.
- The *concluded processing time rule* is based on requested robustness and concluded processing time of the task, $idle_j = f(p_{ij})$.
- The *expected processing time rule* is based on requested robustness and expected processing time of the task, $idle_j = f(p'_{ij})$.

Algorithm 1 Process of Robust Schedule Preparation

```
OPEN and CLOSE are empty
copy ∀t_j from JOB to OPEN
for ∀t_j ∈ OPEN do
  setup S_j to S_JOB
end for
repeat
  for ∀t_j ∈ OPEN do
   for ∀t_k, where t_k takes precedence over t_j do
     if ∀t_k ∈ CLOSE then
      setup S_j to max(∀C_k'; t_k takes precedence over t_j)
      move t_j to OPEN'
     end if
   end for
  end for
  use priority rule to select t_NEXT from OPEN'
  use negotiation mechanism to get offers for of C_ij,
where t_j = t_NEXT and ∀i are machines able to process t_j
  use reputation model to get C_ij'
  select C_j' = min(∀C_ij)
  insert idle time to increase robustness
  move t_j from OPEN' to CLOSE
until OPEN' is empty
C_JOB = max(C_j'; ∀j, t_j ∈ CLOSE)
```

Formal description of process of preparation robust schedule preparation is presented by Algorithm 1**¡Error! No se encuentra el origen de la referencia.**. In the algorithm, the S_{JOB} is the starting time of whole operation, the C_{JOB} is its completion time.

4 Experiments

Several configurations of the algorithm were used; each configuration was run 20 times. In all of them simple CNP is used as a negotiation mechanism and 10 machines with limited resources shared across all tasks is involved, their capabilities vary from 70% to 90%. Each machine is able to process any task. Requested robustness is 5%. Number of tasks is counted as square of numbers from 1 to 7. For each task, its resources requirements are equal to 10 times the number of tasks IDs. Three different workflow structures (see Figure 1**¡Error! No se encuentra el origen de la referencia.**) defined by precedence constraints are used:

- *Single string*, all the tasks are in one string, which is a critical path. All priority rules provide the same t_{NEXT}.

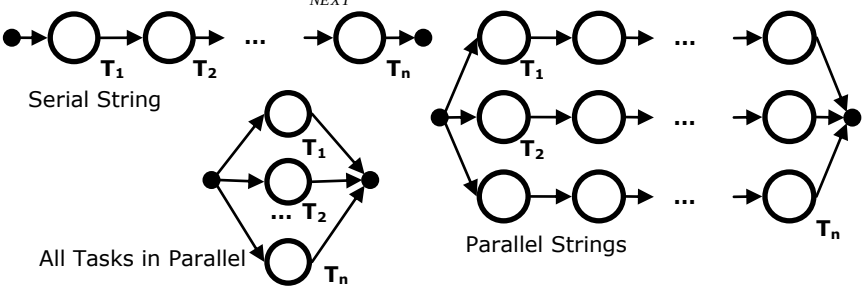

Fig. 1. Applied workflow structures

- *All tasks in parallel*, each parallel string consists of one task. The critical path consists of the longest task(s). "Any task" priority rule provides random t_{NEXT}, the other rules provide the same one.
- *Parallel strings*, all parallel strings consist of same number of tasks and have the same starting date. The tasks are distributed by rule that the following task is added to the tail of the next string. Each of priority rule may provide different t_{NEXT}.

4.1 Results

Example of simulation runs is presented on Figure 2. The set of graphs presents makespan, robustness and scheduling order for number of tasks from 1 to 49 for parallel strings workflow structure and the priority rule of "Any task". The reputation is not assumed; no extra idle times are added.

In most of the results (not just this one), there is a break in results at position between 9 (=3^2) and 16 (=4^2) tasks. It is due to number of tasks that is higher than number of machines; the machines may be overloaded by already scheduled tasks.

The Table 1 presents selected results of simulations for 9 (maximum number of tasks, which is less then number of machines) and 49 tasks (maximum number of tasks in experiments, machines are overloaded). In all simulations the requested robustness was kept. The slack reserves were mainly in parallel structure on non-critical tasks if the priority rule was "Any task". For the linear-parallel structure, there is space for post-processing negotiation with partners about redistribution of slacks on the end of non-critical chains among their tasks. Also it is clear that assuming any information about the future tasks is better than scheduling any task of the ready ones. The average makespans does not differ significantly although the order of tasks scheduling varies, the main difference is in variances of robustness mainly when the reputation models are applied (not in scope of this paper).

References

1. Bíba, J., Vokříınek, J., and Hodík, J. Renegotiable competitive contract net protocol. Technical Report GL 194/08, ATG, Gerstner Laboratory, CTU, 2008.
2. Davenport, A. J., Gefflot, C., and Beck, J. C. Slack-based techniques for robust schedules. In Proc. of the Sixth European Conf. on Planning (ECP-2001), 2001.
3. Kouvelis, P. and Yu, G. Robust Discrete Optimization and Its Applications. Springer, 1996.
4. Pape, C. L. Constraint propagation in planning and scheduling. Technical report, Technical Report, Stanford University, Palo Alto, California, 1991.
5. Perugini, D., Jarvis, D., Reschke, S., and Gossink, D. Distributed deliberative planning with partial observability: Heuristic approaches. In Proc. of the Int. Conf. on Integration of Knowledge Intensive Multi-Agent Systems. KIMAS 2007, 2007, 407–412.
6. Pinedo, M. L. Planning and Scheduling in Manufacturing and Services. Series in Operations Research and Financial Engineering. Springer, 2006.
7. Vokřínek, J., Bíba, J., Hodík, J., Vybíhal, J., and Pěchouček, M. Competitive contract net protocol. In van Leeuwen et Al., J., editor, SOFSEM 2007: Theory and Practice od Computer Science, Berlin. Springer, 2007a, 656–668.
8. Vokřínek, J., Bíba, J., Hodík, J., Vybíhal, J., and Volf, P. RBVO formation protocol. In 2007 IEEE/WIC/ACM International Conference on Intelligent Agent Technology (IAT 2007), 2007b, 454–457.

Table 1. Selected simulation results

		Scheduled makespan (No. of tasks):			
Average robustness based on processing time:		**9** (more machines than tasks)		**49** (more tasks than machines)	
Structure:	**Priority rule:**	**Nominal**	**Concluded**	**Nominal**	**Concluded**
Linear	Any task	1.12	1.18	1.11	1.17
Parallel	Any task	1.21	1.25	11.89	12.68
	CP nominal	1.11	1.56	14.74	16.13
Linear-parallel	Any task	1.12	1.17	1.21	1.28
	CP nominal	1.12	1.17	1.11	1.17
	CP current	1.12	1.17	1.11	1.17
	Longest successors	1.12	1.17	1.12	1.17

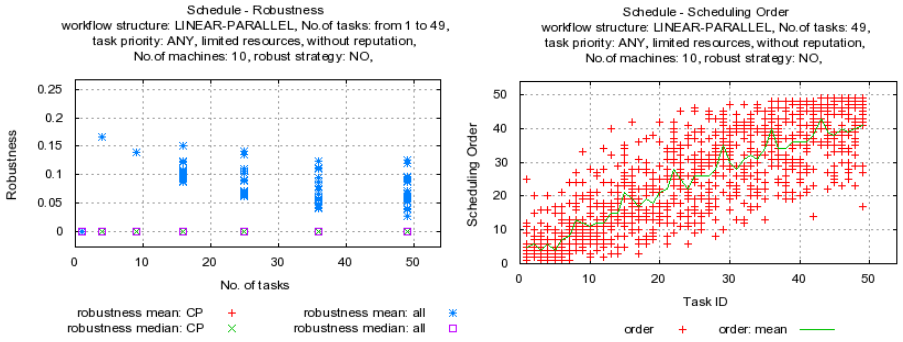

Fig. 2. Example of simulation results