NRC·CNRC

NRC Publications Archive
Archives des publications du CNRC

An interactive decision support method for multi-project schedule coordination

Hao, Q.; Wang, S.; Xue, Y.; Shen, W.

This publication could be one of several versions: author's original, accepted manuscript or the publisher's version. / La version de cette publication peut être l'une des suivantes : la version prépublication de l'auteur, la version acceptée du manuscrit ou la version de l'éditeur.

Publisher's version / Version de l'éditeur:

CSCE 2009 Annual General Meeting & Conference [Proceedings], pp. 1-3, 2009-05-27

NRC Publications Archive Record / Notice des Archives des publications du CNRC :
https://nrc-publications.canada.ca/eng/view/object/?id=2b4d469d-9ed8-46ce-a643-89dae3ce7635
https://publications-cnrc.canada.ca/fra/voir/objet/?id=2b4d469d-9ed8-46ce-a643-89dae3ce7635

Access and use of this website and the material on it are subject to the Terms and Conditions set forth at
https://nrc-publications.canada.ca/eng/copyright
READ THESE TERMS AND CONDITIONS CAREFULLY BEFORE USING THIS WEBSITE.

L'accès à ce site Web et l'utilisation de son contenu sont assujettis aux conditions présentées dans le site
https://publications-cnrc.canada.ca/fra/droits
LISEZ CES CONDITIONS ATTENTIVEMENT AVANT D'UTILISER CE SITE WEB.

Questions? Contact the NRC Publications Archive team at PublicationsArchive-ArchivesPublications@nrc-cnrc.gc.ca. If you wish to email the authors directly, please see the first page of the publication for their contact information.

Vous avez des questions? Nous pouvons vous aider. Pour communiquer directement avec un auteur, consultez la première page de la revue dans laquelle son article a été publié afin de trouver ses coordonnées. Si vous n'arrivez pas à les repérer, communiquez avec nous à PublicationsArchive-ArchivesPublications@nrc-cnrc.gc.ca.

National Research Council Canada | Conseil national de recherches Canada

Canada

NRC·CNRC Institute for Research in Construction
CNRC·NRC Institut de recherche en construction

http://irc.nrc-cnrc.gc.ca

An interactive decision support method for multi-project schedule coordination

NRCC-51129

Hao, Q.; Wang, S.; Xue, Y.; Shen, W.

May 2009

A version of this document is published in / Une version de ce document se trouve dans:
CSCE 2009 Annual General Meeting & Conference, St-John's Newfoundland, May 27-30, 2009, pp. 1-3

The material in this document is covered by the provisions of the Copyright Act, by Canadian laws, policies, regulations and international agreements. Such provisions serve to identify the information source and, in specific instances, to prohibit reproduction of materials without written permission. For more information visit http://laws.justice.gc.ca/en/showtdm/cs/C-42

Les renseignements dans ce document sont protégés par la Loi sur le droit d'auteur, par les lois, les politiques et les règlements du Canada et des accords internationaux. Ces dispositions permettent d'identifier la source de l'information et, dans certains cas, d'interdire la copie de documents sans permission écrite. Pour obtenir de plus amples renseignements : http://lois.justice.gc.ca/fr/showtdm/cs/C-42

National Research Council Canada | Conseil national de recherches Canada
Canada

2nd International/ 8th Construction Specialty Conference
2e conférence internationale / 8e conférence spécialisée sur le génie de la construction

St. John's, Newfoundland and Labrador / *St. John's, Terre-Neuve et Labrador*
May 27-30, 2009 / *27-30 mai 2009*

An Interactive Decision Support Method for Multi-project Schedule Coordination

Qi Hao, Shuying Wang, Yunjiao Xue, Weiming Shen
Centre for Computer-assisted Construction Technologies, National Research Council, London, Ontario

Abstract: In large engineering projects, conflicts arise when project activities compete for limited and shared resources. For example, a construction company executes two or more projects with thousands of tasks, located on different sites, competing for the same equipment, tools, materials, and workers. This belongs to the Resource Constrained Multi-Project Scheduling Problem (RCMPSP) involving constraints defined on multiple projects. It is a problem that none of the existing single-project RCPSP approaches can deal with easily. The key to coordinating multiple projects is to detect and resolve the conflicts through a decision making process. This paper proposes an interactive decision support method to achieve the balancing of multiple objectives – time, resources, and performance for the coordination of multiple concurrent projects. Interactive decision support means detecting conflicts, prioritizing projects, proposing potential options, as well as calculating the ripple effects of each step after a decision is made. The proposed approach and related algorithms have been fully implemented and tested in a web-based aircraft periodical inspection and maintenance system and are being applied in other ongoing projects for multi-project coordination in construction and facilities management.

1. Introduction

Most research efforts in project scheduling tend to focus primarily on single project scheduling. This problem is known as the resource-constrained project scheduling problem (RCPSP) and is considered NP-hard in the strong sense (Blazewicz et al. 1983). However, it is essential to extensively cover the multiple project scheduling problems because most of the real life projects involve global resource constraints and the launching of concurrent projects in order to effectively utilize limited resources. The Resource Constraint Multiple Project Scheduling Problem (RCMPSP) is an extension of the well-known RCPSP problem and it involves the precedence constrained scheduling of two or more projects' activities competing for the same set of scarce resources (Katsavounis 2008). RCMPSP is an over-determined problem involving conflicting constraints defined in multiple projects and it is a problem that none of the existing single-project RCPSP approaches can deal with easily.

In large engineering projects in the construction and manufacturing sectors, conflicts arise when project activities compete for shared resources. For example, a construction company executes two or more concurrent projects that contain thousands of tasks, located on different sites, competing for the same equipment, tools, materials, and workers. The key to schedule coordination of multiple projects is to detect conflicts and resolve the conflicts through a decision making process. In general, the basic decision options are: "prioritizing" (projects or tasks), "crashing" (tasks), "shifting" (tasks), and "releasing" (constraints). Whatever the options and choices would be, a manager feels hard-pressed to take actions because one cannot foresee the ripple effects of the choices in the long run. An accurate estimation of the impact on one project is hard to tell, not to mention the effect on other projects. Without accurate

information and prediction of the ripple effect, the decisions are made blindly based on past experience and subjective judgement. To assist human decision makers in project coordination processes, this paper proposes the use of an interactive decision support method to gain the balancing of multiple objectives – time, resources, and performance for the coordination of multiple concurrent projects. Interactive decision support means detecting conflicts, prioritizing projects, proposing potential options, as well as calculating the ripple effect of each step after a decision is made.

This paper describes the RCMPSP problem based on the requirements arising from aircraft inspection and maintenance practices. The interactive decision support method is then proposed in which the coordination of multiple aircraft inspection projects is achieved through a number of novel algorithms for the detection and resolution of conflicts. The ripple effect of a decision is calculated simultaneously on involved projects using a heuristic-based scheduling algorithm, based on the dynamically generated partial task networks. The proposed technology and algorithms have been fully implemented and tested in a web-based aircraft periodical inspection and maintenance system. Some of the implementation results are briefly presented in the paper.

2. Research Literature

Plentiful approaches or solutions have been proposed in the literature to solve the RCPSP problem using mathematical modeling (including integer programming, dynamic programming, and branch-and-bound approaches), constraint satisfaction, heuristics and meta-heuristics based computation methods. Herroelen et al. (1998) directed readers to several early reviews since the 1960s while the survey itself focused on recent progress made with optimal branch-and-bound procedures and their important extensions. Based on the review and summary of 200 papers, Brucker et al. (1999) proposed a classification scheme according to scheduling environments, activity characteristics, and objective functions to achieve a common notation and classification scheme in the project scheduling domain. Kolisch and Hartmann looked after the heuristic approaches that they thought were more practical and feasible in solving real-world RCPSP problems (Hartmann and Kolisch 2000; Kolisch and Hartmann 2006). Sriprasert and Dawood (2003) developed a Lean Enterprise Web-based Information System that addressed some special needs for project management in construction: multi-constraint information management, visualization, 4D modeling and simulation, mobile data retrieving and data collection, etc.

In terms of multi-project scheduling, only a few researches have focused on RCPSP problems in the scope of multiple projects. Because of the complexity of RCMPSP problems, priority rule-based heuristics become the only feasible way to construct a feasible algorithm. Katsavounis (2008) formulated multiple projects as a single-entry single-exit weighted directed acyclic graph and applied a single-pass, parallel scheduling heuristic on top of the standard critical path calculation of each individual project. This heuristics-based approach was proved on a small test base with 3 concurrent projects and 9-12 tasks in each project. Khattab and Soyland (1996) believed priority-based rules outperform CPM-based rules used in commercial packages (i.e. Primavera^{TM}) in terms of levelling limited resources among multiple construction projects. Meta-heuristic technologies (for example, genetic algorithm models in Liu et al. 2005) are also applied to the multi-project scenario, where multiple projects' schedules are achieved by a combined global objective function above the project level.

A number of researchers have proposed the use of distributed intelligence (specifically, agent technology) to handle the complexity of the RCMPSP problem. Li and Liu (2005) developed a multi-project planning and scheduling system using a distributed agent-based approach. Each project in the proposed multi-agent framework is presented physically by a project agent. A negotiation-based planning and control mechanism is developed to coordinate these distributed project agents. However, global shared resources are not addressed; rather, the resources are scheduled and balanced within each project. Brown and McCarragher (1999) proposed a negotiation-based distributed resource conflict resolution approach between maintenance agents, process units and other entities in order to coordinate maintenance and production processes in a manufacturing environment. Results have shown reductions in conflict of over 60% compared to a fixed maintenance schedule. DISA (Distributed Interactive Scheduling with Abstractions) (Berry et al. 1994) employed a dynamic multi-agent architecture to address the uncertainties in real-world domains. Temporal abstractions, in the form of summarizations and

generalizations, are applied to agents in different hierarchies for problem reduction and conflict resolution. The system involves the human-in-the-loop through interactive user interfaces and user interactions. The interactive decision support method proposed in this paper advocates the involvement of human in the decision making process, too. Compared with Berry's approach, our solution is based on practical heuristics for conflict detection, project prioritization and conflict resolution; it has no strictly divided hierarchical temporal abstractions and severance of functions between long-term, middle-term and short-term scheduling horizons.

3. Problem Specification

One of the applications of RCMPSP is aircraft inspection and maintenance. In order to maintain an aircraft in a state of "airworthiness", regulations require various kinds of periodical inspections. In aircraft periodical inspections, major scheduled inspection tasks need to be carried out sequentially on an aircraft that is temporarily taken off its missions. The inspection tasks for a fleet of aircraft require extensive expertise in practice, constant re-assessment of changing priorities, and frequent re-scheduling in response to changes in personnel, skill sets, and equipment availability.

Conflicting schedules caused by shared resources or other constraints across projects are the major issue to be resolved in order for the whole schedule to be practical. The RCMPSP in aircraft inspection and maintenance is very complex in that:

- A regular project contains thousands of tasks depending on the size of the projects and areas of applications.
- Several projects often claim the use of common and limited resources, which are defined as resource constraints. If concurrent tasks (in different projects) that require the same resource are scheduled to roughly the same time frame, the system needs to find a way to serialize them according to their priorities. The following are some examples of resource constraints:
    - Shared equipment and tools
    - Shared staff with different qualifications
    - Shared working spaces with limited access capacity
- There are customizable exclusive constraints. Theoretically, tasks can be executed simultaneously when they do not have precedence relationships and they are not competing for a common resource. However, it is possible that these tasks are still "exclusive" in nature so that they cannot be preceded at the same time. For example, a painting job on an aircraft requires that, within the whole maintenance facility, all other scheduled tasks which require electricity must be suspended until the painting job is finished. In other words, a painting job might have an impact on the time schedules of all ongoing aircraft inspection projects.
- Coordination of multiple projects is required because of global exclusive constraints and shared resource constraints. The focus of this paper is conflict detection and resolution algorithms for multi-project scheduling coordination.
- Dynamic re-scheduling needs to be carried out frequently within- or across- projects.

4. Definition of the RCMPSP Problem

4.1 Task networks

For each given project, we describe it as a task network. A task network $N = (T, P)$ is composed of a set of tasks $T = \{t_1, t_2, \ldots, t_n\}$ and a set P of precedence relationships between tasks, where n is the number of tasks. Among them, t_1 and t_n stand for two dummy tasks which are the start and the finish of the project. A task $t \in T$ contains the following attributes:

- ID - an unique identifier
- D - duration
- EST - earliest start time
- LST - latest start time
- EFT - earliest finish time
- LFT - latest finish time

- Priority - task priority is an integer between 1 and 9.
- Critical - a critical task is a task $t \in T$, if *t.EST = t.LST*, or the slack between *t.EST* and *t.LST* is zero. The delay of a critical task will cause the delay of the entire project. The paths composed by critical tasks are called "Critical Paths".
- Split-able - this Boolean value specifies whether the duration of this task is allowed to be split or not. For example, a non-split-able task has to be arranged in a continuous time period (even when its duration spans over silent or overtime hours). A task is "split-able" by default.

A multi-task network is a set of task networks represented by $MTN = \{N_1, N_2, ..., N_M\}$, where *M* is the number of task networks and each network is independent from each other.

4.2 Constraints

A task $t \in T$ in network *N* contains the following constraints:

- Precedence constraints - ***P***
  Given a task network *N = (T, P), f*or *pt*, *st*$\in T$, (*pt*, *st*)$\in P$ means that *pt* is a preceding task of *st* and *st* is a subsequent task of *pt*; or *st* cannot start if *pt* is not finished. Meanwhile, a task cannot start until all its preceding tasks are finished. For a task $t \in T$, we use *Pre(t)* and *Sub(t)* to denote all direct preceding tasks and direct subsequent tasks of *t* in *N*. Formally, we have
  pt$\in$*Pre*(*t*)$\Rightarrow\exists$*pt*$\in T$: $\exists$(*pt*, *t*)$\in P$, and *st*$\in$*Sub*(*t*)$\Rightarrow\exists$*st*$\in T$: $\exists$(*t*, *st*)$\in P$.
- Resource constraints - ***RC***
  Suppose that there are *K* types of renewable resources and the resource constraints applied to a task *t* can be represented as $(t, r_k, \delta_{tk}) \in RC, 1 \le k \le K$, where r_k is a type of resource and δ_{tk} is the capacity that is required by task *t* on r_k. A resource r_k is available with a limited capacity a_k at any renewable time period ζ. The constraint that a resource r_k put on the whole project is that at any time, the requirement of all tasks to the resource should be within its maximum capacity/availability.
  $$\sum_{t_i \in T} \delta_{t_i k} \le a_k, i = 1,2,...,n; k = 1,2,...,K$$
- Exclusive Constraint - ***EC***
  Suppose that there are *L* types of exclusive requirements, an exclusive constraint applied to a task $t \in T$ can be represented as $(t, e_l) \in EC, 1 \le l \le 1$, where e_l is an exclusive constraint. It must be satisfied that for the whole project, if a task t has an exclusive constraint e_l defined on it, then for $\forall t' \in T$ and $t' \ne t$:
  $$(t'.EFT \le t.EST) \cup (t' EST \ge t.EFT)) \cap ((t'.LFT \le t.LST) \cup (t'.LST) \ge t.LFT)$$
  Depending on the scale of its impact, an exclusive constraint could be "Local" - *LEC* or "Global" - *GEC*. The former refers to a type of exclusive constraint that will affect the tasks within the same task network only; while the impact of the second type of exclusive constraint will expand to affect all tasks within the *MTN*. The conflict resolution method presented in this paper only considers the *global* conflicts caused by *GEC*s. We define a task that has an exclusive constraint defined on it as an "Exclusive Task". The effect of the exclusive task on other tasks is described as a set of "Affected Tasks" which belong to either the same project or other projects. By default, all tasks are affected by an exclusive task unless explicitly specified.

4.3 Active tasks

A time window *TW* is a time period defined by a lower time bound and an upper time bound, i.e. *TW* = $[LowerT, UpperT]$. Active tasks (*AT*) are a set of tasks that have either start time or end time or both start time and end time falling into or coving the time window.

AT = {t_1, t_2, …, t_x} $\exists t_i \in MTN, 1 \le i \le x$ and

$$((LowerT \le t_i.EST) \cap (t_i.EFT \le UpperT)) \cup ((t_i.EST \le LowerT) \cap (t_i.EFT \le UpperT)) \cup$$
$$((t_i.EST \ge LowerT) \cap (t_i.EFT \ge UpperT)) \cup ((LowerT \ge t_i.EST) \cap (t_i.EFT \ge UpperT))$$

4.4 Conflicts

A conflict *cf* is defined by the following five attributes:

- ID - an unique identifier
- CauseTask – defined by a task that has a global exclusive constraint or a resource constraint
- AffectedTaskList - a task list which contains all the tasks in other projects that will be affected by the *CauseTask* at the same time period.
- Type – conflicts may be caused by two types of constraints: "exclusive" (*GEC*) or "resource" (shared resource constraints).
- Status - the status of conflict, either "solved" or "unsolved"

Actually, a *cf* stands for a group of conflicts that is caused by a common *CauseTask*. It includes the pair-wised conflicts between the *CauseTask* and each of the affected tasks on the *AffectedTaskList*.

4.5 Overlap tolerance rate

Depending on the scale of overlap situations, some minor schedule overlaps can be ignored in order to reduce the number of conflicts. This is set through the overlap tolerance rate *otr* representing the percentage of overlap allowed between conflicting tasks. The *overlap tolerance rate* on a task is calculated by comparing the time schedule of the task with that of its conflicting cause task. For a given exclusive task *t1* and an affected task *t2*, the overlap function *OTR(t1,t2)* is defined to be 1 when *t1* fully covers, equals, or is covered by *t2*; or it is defined to be the quotient of the overlapping time period over *t1.duration* in other occasions.

5. Algorithms for Multi-project Schedule Coordination

The assumptions for multi-project coordination are: 1) all projects involved have already been re-scheduled starting from the same time point; and 2) all conflicts within a single project have already been resolved using the dynamic task network scheduling algorithm (Hao et al. 2008). This means that before entering the multi-project coordination procedure, the schedule of each project is good for execution without considering other projects.

Our solution for multi-project coordination can be broken into several tasks: to detect conflicts for the given projects in a given time window; to solve the identified conflicts according to the priority rules or the modified task attributes; to create updated project schedules; and to assist the decision making process by analysing the impact of decisions and comparing different trials. In fact, for the same scenario, users can apply different strategies to resolve conflicts and get different resolution results. We call the whole decision making process a "trial"; and a trial contains all the information of active tasks, conflicts, strategies/rules, partial resolution options and results and final solutions.

5.1 Conflict Detection

A time window is required to detect schedule conflicts between multiple projects. The system will then try to detect and resolve the conflicts between tasks that fall within this time window. There are two reasons to define a time window.

1) For large scale projects, multiple projects planned in the same time horizon may have a total number of tens of thousands of tasks. Finding and resolving the conflicts in this huge search space is uncontrollable. Using a time window to reduce the search scope and reduce the number of conflicts is a practical and an efficient approach.

2) In a dynamic scheduling environment, the status of a schedule is constantly changing with the availability of limited resources and the actual execution data of tasks. Re-scheduling a project is a frequent practice that is necessary for a well-managed organization. Similarly, the coordination of multiple projects in their full life span is not useful since everything is changing frequently. The parallel projection for tasks outside the time window is a reasonable simplification of the problem.

```
INPUT : AT, otr
OUTPUT : CL          // conflict list
BEGIN :
    // el : exclusive task list, rl : resource task list
    CL = ∅;  el = ∅; rl = ∅;
    if (AT ≠ ∅){
        el = {∃t ∈ AT, (t, e_l) ∈ EC, 1 ≤ l ≤ L};
        rl = {∃t ∈ AT, (t, r_k, δ_tk) ∈ RC, 1 ≤ k ≤ K};
    }
    if (el ≠ ∅){
        for(∃t ∈ el){
            cf = ∅;
            for(∃t' ∈ AT, t' ≠ t){
                if(OTR(t, t') ≥ otr)
                    cf.affectedtasklist = cf.affectedtasklist ∪ t';
            }
            if(cf.affectedtasklist ≠ ∅){
                cf.causetask = t;
                cf.status = 'unsolved';
                cf.type = 'exclusive';
            }
            CL = CL ∪ cf;
        }
    }
    // resource conflict detection is samilar with exclusive conflict, but
    // needs to compare the resource constraints of two tasks;
    ...
    // sort conflicts according to the EST of exclusvie task in a conflict.
    if (CL ≠ ∅)
        sortConflictlistByTime(CL);
END.
```

Figure 1: Detect conflicts in a given time window

The algorithm to detect conflicts for a given time window is described in Figure 1. In this algorithm, we scan the set of active tasks to find all the tasks that have exclusive constraints (*GEC*) or resource constraints defined on them and separate these tasks into two lists: *el* for exclusive tasks and *rl* for resource tasks. For each exclusive task $t \in el$, all tasks $t' \in AT, t' \neq t$ are checked if they are conflicting with *t* during the time period of [*t.EST, t.EFT*]. We calculate the list of affected tasks of the conflict (*cf.affectedTaskList*) by selecting those overlapping tasks that have an *OTR* greater than the number specified by *otr*. After that, if the affected task list is not empty, we save it as a conflict and append this conflict to the conflict list *CL*. Similarly, the resource-type conflicts are calculated and added to the conflict list as well. Finally, all the conflicts in the *CL* are sorted according to their time sequence.

5.2 Conflict Resolution

Conflict resolution is done through an iterative decision making process involving the human-in-the-loop. Theoretically, the target of conflict resolution is to "serialize" the conflicting tasks in time. Conflicts are resolved one after another by applying a set of default or selected priority rules; either allowing the system to solve all the conflicts or involving human interactions in every step. After a group of conflicts caused by the same *CauseTask* is resolved, conflicts are generally shifted so the system needs to detect conflicts again in order to solve remaining conflicts. This is an iterative process until all conflicts are resolved.

5.2.1 Priority rules

Different priority rules can be used to solve the schedule conflicts of overlapping tasks. By default, an exclusive task always has a higher priority than other tasks. However, when two exclusive tasks are in conflict, the system has to apply certain rules (default or user-selected) to decide the priorities of the two tasks.

The user can choose one or more of the following rules and apply them to the conflict resolution process:

- An exclusive task has always a higher priority than a non-exclusive constraint task.
- For any two exclusive tasks, *t* and *t'*, or when any two tasks which require the same resource are in conflict, the following rules can help the user decide if *t* has a higher priority.

- Closest To Project End Date: $(t_n.EFT - t.EFT) < (t_n'.EFT - t'.EFT)$
- Task Priority: *t.priority>t'.priority*
- Earliest Planned Start Time First: *t.EST>t'.EST*

5.2.2 Adjustment of task attributes

In addition to the above conflict resolution rules, the user can indirectly solve or shift a conflict by changing the attributes of the critical tasks. The following are some strategies for changing a task's attributes:

- "Crushing" the task duration. Squeezing the duration of critical tasks (for example, an exclusive task) can significantly reduce the number of conflicts in the conflict list and thus reduce its impact on other projects.
- "Shifting" tasks. Changing the earliest start time of a task (*t.EST*) can move a task to another non-conflicting time slot so when the system re-checks the conflicts, the conflict list shifts, too.
- "Releasing" task constraints. Removing an unnecessary *GEC* definition of an exclusive task, or changing the resource requirement of a task to an alternative source, can release constraints.
- "Prioritizing" tasks. For individual tasks, this is done very obviously through changing *t.priority*. However, prioritizing tasks is usually done through the priority rules described in 5.2.1.
- Overtime arrangement. By changing a task to a non-split-able task, or opening up an additional work shift, a user can make overtime arrangements for critical tasks so that the impact of these tasks can be minimized.

5.2.3 Solving conflicts and updating the schedule

After selecting a priority rule or changing a task's attributes, the system needs to update the schedules for all relevant projects. Figure 2 shows an example of schedule updating for three projects during a conflict resolution process. In this example, an exclusive conflict *cf* is defined as <'*cf001', t2, {t5, t6, t7, t9, t10}, 'exclusive', 'unsolved'*>, where: '*cf001*' is the identification; *t2* is an exclusive task; {*t5, t6, t7, t9, t10*} is the list of affected tasks that are in conflict with task *t2*; 'exclusive' means that '*cf001*' is an exclusive conflict; and '*unsolved*' shows the status of '*cf001*'; it is not solved yet. In STEP 1, as both *t2* and *t5* are both exclusive tasks, the system (or the user) has to decide which task has a higher priority using the rules set in 5.2.1. If *t2* has a higher priority, the system will set the ESTs and EFTs of all the affected tasks (including tasks *t5, t7, t9* and *t10*) to time slots after *t2.EFT* and make updates to each affected project schedule accordingly in STEP 2. The status in *cf* will be changed to '*solved*' after this step. After project schedules are updated, conflicts are re-checked by the system and a new exclusive conflict cf' is identified as <*'cf002', t5, {t3, t6, t7, t9, t10}, 'exclusive', 'unsolved'*>. Since an exclusive task always has higher priority than other non-exclusive tasks, this conflict can be solved as shown STEP 3.

It seems all conflicts are solved after STEP 3. However, since all tasks in the affected projects (*t5* and *t7* Project 2; *t9* and *t10* in Project 3) are moved backwards, the impact of a conflict resolution process on the related projects is always "negative" - postponing of project schedules. This is a situation that a manager never wants to see. In reality, any decision made is a trade-off between positive and negative results.

The final step justifies a valuable solution in the conflict resolution process by bringing in "positive" effects to project schedules. As shown in STEP 4, the system does a final check and moves a short task *t7* forward in Project 2. In detail, when updating Project 2's schedule, a virtual task *vt* is added before *t5* and *t7* in the partial network and *vt.EST = vt.EFT = t4.EFT*. The system finds *t7* could fill the time gap between *vt* and *t2* because *t7.EST >= vt.EFT* and *t7.EFT <= t2.EST* and sets *t7.EST = vt.EST = t4.EFT*.

The calculation of an updated project schedule is done through creating a partial task network from the multiple independent heads (for example *t5* and *t7* in STEP 2 but not *t6*) to the final task in the task network. Dynamic scheduling based on a partial task network is described in a separate paper (Hao et al. 2008). The reason for using the partial network approach is that if the expected time schedule for one task is changed, all the direct and indirect subsequent tasks of this task should be changed; while the preceding tasks of this task and other tasks that are neither a preceding nor subsequent task of this task are not affected. By excluding the tasks that should not be changed, the cost of computation is significantly reduced.

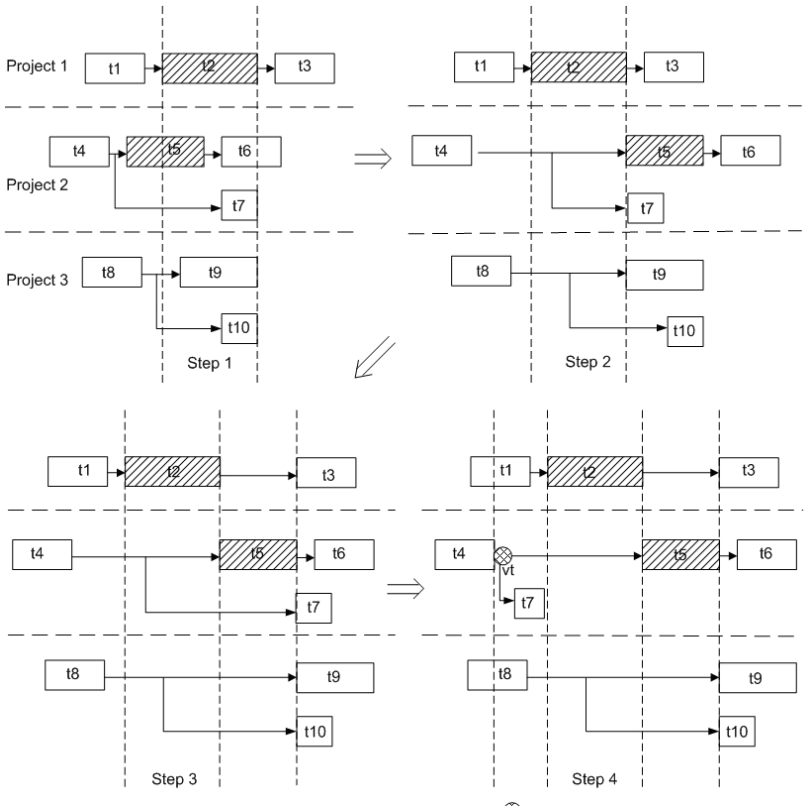

Figure 2: Conflict detecting, resolution and schedule updating

5.3 Impact Analysis

Conflict resolution is an iterative process to detect conflicts, resolve conflicts and update schedules until all conflicts are resolved. The purpose of impact analysis is to provide meaningful support after every decision to solve a conflict or to change task attributes. Impact analysis provides a reasonable time projection of all related projects in scope. The impact on the project schedule is the most valuable data for decision support especially when thousands of tasks from different projects are taken into consideration. Through impact analysis, the user can evaluate whether he/she has made a good decision in the previous step. The impact on a project can be a positive or negative value calculated by $t_n EFT' - t_n EFT$, where $t_n EFT'$ is the newly calculated project finish time and $t_n EFT$ is the project finish time of the previous step or the original project finish time.

5. Implementation and Case Studies

We have implemented the concepts and algorithms proposed in our work to address the RCMPSP problem in the aircraft inspection and maintenance domain. A fully functional web-based system was developed to manage aircraft inspection projects. The system supports creating new projects from pre-defined templates. The inspection tasks included in the templates are usually routine tasks that must be applied to every aircraft under inspection. A sample template contains 700 to 800 routine inspection tasks and 600 to 700 dummy tasks for marking the boundaries of 12 major inspection groups (“phases”) and roughly 300 lower-level inspection groups (or “cards”) in the project hierarchy. Users can dynamically add new inspection tasks during any stage of a project.

When creating a new project, the system requires the project manager to provide a planned start time. This given time will be used to create the first plan for the project (this plan can be used as a baseline for

project performance review). Then, the user can request to re-plan a project whenever it is necessary. The coordination of multiple projects starts with an interface to select projects in scope and input a time window. The system then comes out with a list of conflicts detected (Figure 3). Conflicts are organized into groups where each group has the same *CauseTask* (the highlighted tasks). Each task in the conflict list is provided with a link as a quick reference to modify its attributes. The conflicts can be resolved by the system automatically or by the user interactively. Also the impact can be seen in each step (marked "1") or the summary of impacts can be viewed at the end (marked "2").

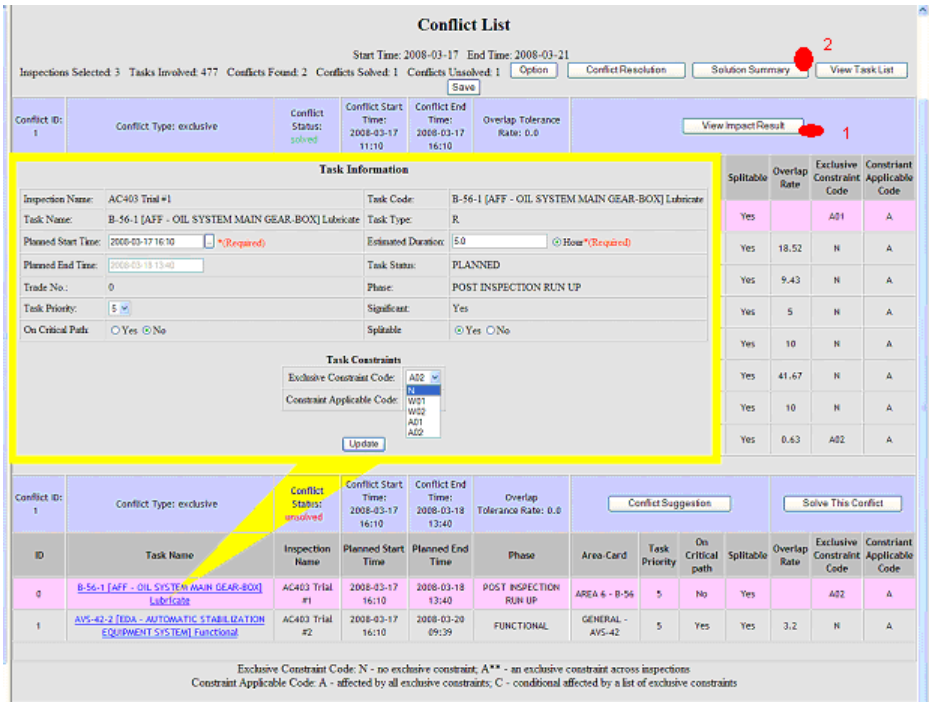

Figure 3: The screen shot of the conflict list

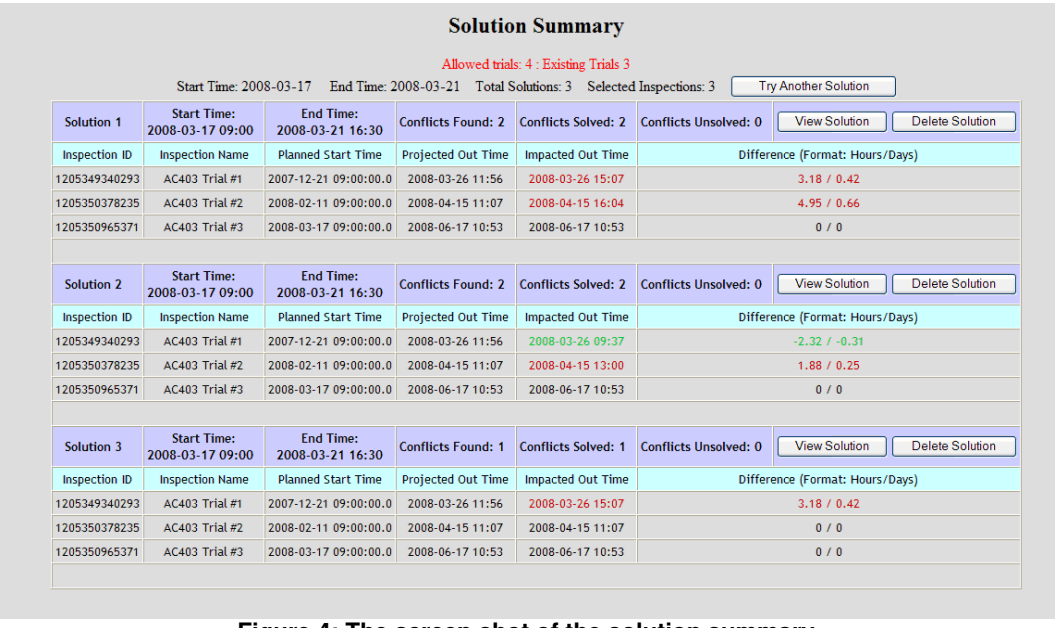

Figure 4: The screen shot of the solution summary

Figure 4 shows the summary of the conflict resolution results after three trials. An ideal system needs to record all status information along with the decision making process, including the transient partial

networks. However, this requires explosive usage of system memories and unmanageable calculation time. In our algorithms, we provide impact analysis in each trial by recording all related information but the partial networks, including conflict lists, task lists, rules and the resulting impact. In the solution summary page, the user can 1) open up another new trial ("Try Another Solution") for the same scenario; 2) select and view a solution in detail; 3) delete a solution from the memory. During a conflict resolution period, the user can also switch from one trial to another to compare the results of different solutions and select the best solution as their final decision.

6. Conclusion

The resource constrained multiple project scheduling problem (RCMPSP) is a NP-Hard problem in a strong sense. In large manufacturing or construction projects, the objective is to find a feasible, rather than optimal solution, within a reasonable computation time using rule-based heuristics. Our solution for multi-project coordination can be broken into several tasks: to detect conflicts for the given projects in a given time window; to solve the identified conflicts according to the priority rules or the modified task attributes; to create updated project schedules; and to assist the decision making process by analysing the impact of decisions and comparing different trials. The proposed concepts and algorithms have been fully implemented in a web-based aircraft periodical inspection and maintenance system. The prototype system is ready for field testing and deployment.

Reference

Berry, P.M., Choueiry, B.Y. and Friha, L. 1994. Distributed approach to dynamic resource management based on temporal influence. *Intelligent Systems Engineering* 3(2):79-86.

Blazewicz, J., Lenstra, J.K., and Rinnooy Kan, A.H.G. 1983. Scheduling subject to resource constraints: classification and complexity. *Discrete Applied Mathematics* 5:11-24.

Brown, J. and McCarragher, B.J. 1999. Maintenance resource allocation using decentralized co-operative control. Proceedings of the Information, Decision and Control. *Data and Information Fusion Symposium, Signal Processing and Communications Symposium and Decision and Control Symposium*, 41-6.

Brucker, P., Drexl, A., Möhring, R., Neumann, K. and Pesch, E. 1999. Resource constrained project scheduling: notation, classification, models and methods. *European Journal of Operational Research* 112:3-41.

Hao, Q., Xue, Y., Wang, S. and Shen, W. 2008. A dynamic scheduling algorithm for time- and resource-constrained task networks. To be submitted to IEEE SMC 2009.

Hartmann, S. and Kolisch, R. 2000. Experimental evaluation of state-of-the-art heuristics for the resource-constrained project scheduling problem. *European Journal of Operational Research* 127(2):394-407.

Herroelen, W., De Reyck, B. and Demeulemeester, E. 1998. Resource-constrained project scheduling: a survey of recent developments. *Computers and Operations Research* 25(4): 279–302.

Katsavounis, S. 2008. Scheduling multiple concurrent projects using shared resources with allocation costs and technical constraints. *Proceedings of the Third IEEE International Conference on Information and Communication Technologies from Theory to Applications,* 1-6.

Khattab, M.M. and Soyland, K. 1996. Limited resource allocation in construction projects. *Computers and Industrial Engineering* 31(1-2):229-32.

Kolisch, R., Sprecher, A. and Drexl, A. 1995. Characterization and generation of a general class of resource-constrained project scheduling problems. *Management Science* 41:1693-1703.

Li, J. and Liu, W. 2005. An agent-based system for multi-project planning and scheduling. *Proceedings of the IEEE International Conference on Mechatronics and Automation*, 2:659-64.

Liu, T., Liu, M., Zhang, Y. and Zhang, L. 2005. Hybrid genetic algorithm based on synthetical level of resource conflict for complex construction project scheduling problem. *Proceedings of the IEEE International Conference on Machine Learning and Cybernetics*, 9:5699-703.

Sriprasert, E. and Dawood, N. 2003. Multi-constraint information management and visualisation for collaborative planning and control in construction. *ITcon*, 8:341-366.