http://ansinet.com/itj

ITJ

ISSN 1812-5638

INFORMATION TECHNOLOGY JOURNAL

ANSInet
Asian Network for Scientific Information
308 Lasani Town, Sargodha Road, Faisalabad - Pakistan

Information Technology Journal 11 (7): 829-839, 2012
ISSN 1812-5638 / DOI: 10.3923/itj.2012.829.839

The Scheduling Problem of Active Critical Chain Method

W.L. Peng and H. Xu
School of Economics and Management, Shenyang Ligong University, No.6,
Nanpingzhong Road, New Hunan District, Shenyang, 110159, China

Abstract: Critical chain method requires the non-critical activities to be executed as late as possible. However, in some project cases, such as software development projects, the critical chain method might result in the higher risk of project delay. In this study, we present a revised critical chain method for the projects where all the activities should be scheduled as early as possible. The revised critical chain method, defined as the Active Critical Chain Method (ACCM), is generated based on the active baseline schedule, where all activities are scheduled as early as possible. The generation procedure of ACCM schedule is illustrated, including searching critical chain and setting buffers. Referring to the conceptual models of the Resource Constrained Project Scheduling Problems (RCPSPs), we propose the scheduling problem of ACCM and formulate it. Since, the scheduling problem of ACCM is a NP hard problem, we resort to heuristics and the Genetic Algorithm (GA) is investigated to solve it. Finally, a full computation experiment is performed to determine the parameter configuration and effectiveness of the algorithm is verified by some modified benchmark instances.

Key words: Critical chain method, project scheduling problem, genetic algorithm

INTRODUCTION

Although resource constrained project scheduling problems have been researched for several decades (Nadjafi and Shadrokh, 2008; Rabbani *et al*., 2008; Shi *et al*., 2011), they have not presented a practical solution to deal with the uncertainties in project scheduling up to now. Since, the Critical Chain Method (CCM) is presented by Goldratt (1997), it has attracted widely attention in application and theory fields due to its advantages of managing project uncertainties with a systematical way. Traditionally, the critical chain method requires those activities not in critical chain should be executed as late as possible and thus it should be created based on the right-justification schedule. While WIP (Work in Process) considered, the critical chain method is suitable for the schedules without lots of uncertainties, such as production plans and project schedules of construction engineering. However, for the projects with a high risk of time overrun, such as soft development projects or product design projects, all the activities should be executed as early as possible to avoid project delay and the traditional critical chain method has no ability to schedule and control them.

In this study, we present a revised critical chain method, in which all activities are executed as early as possible. The revised critical chain method is defined as active critical chain method and accordingly, the schedule generated by active critical chain method is defined as active critical chain schedule as well. On the base of Serial Schedule Generation Scheme (SSGS), we study the schedule scheme of the active critical chain method, in which the processes of searching critical chain, computing and inserting buffers are addressed. And then, a project scheduling problem of ACCM is presented and the conceptual model is formulated. Since, the scheduling problem of ACCM is a NP hard problem and hard to be solved by exact algorithms, we resort to heuristics to solve it and a genetic algorithm is presented.

In his novel, Critical Chain, Goldratt (1997) put forward the TOC (Theory of Constraints) to use in project management. He claimed that project leaders try to protect the performance of each step of the project by adding safety for each activity, most of which is wasted. He concludes that the bottleneck of a project is the critical chain and the activities on the critical chain should be protected by adding safety buffers. Subsequently, the approaches of CCM used in project scheduling have been explored in several articles and books. Rand (2000) studied the procedure of TOC used in project management and discussed the differences between CCM and CPM (Critical Path Method). Leach (2000) provided the critical chain procedure with examples. He also discussed buffer management and suggested the RSEM

Corresponding Author: W.L. Peng, School of Economics and Management, Shenyang Ligong University, No.6, Nanpingzhong Road, New Hunan District, Shenyang, 110159, China
Tel: 86-024-24682231 Fax: 86-024-24682231

(Root Square Error Method) for buffer sizing. Similarly, Newbold (1998) presented the critical chain scheduling mechanism by using examples. He claimed that buffers are a kind of aggregation of the risk encountered along the chain of events feeding them. Accordingly, he investigated the C and PM (Cut and Paste Method) and the RSEM methods for buffer sizing, and argued that in practice there is really no need for a scientific method for buffer sizing and buffer sizes should be adjusted based on intuitive assessment of risk (Newbold, 1998). Wei *et al.* (2002) compared and contrasted the advantages and disadvantages of traditional project management and TOC project management. Steyn (2002) pointed out several applications of TOC to project management, such as project scheduling, resource management in concurrent projects, project cost management and project risk management.

Herroelen and Leus (2001) performed a full factorial experiment to determine the factors that affect the performance of a project. In general, they reported that keeping the critical chain activities in series increases the project make-span while regularly updating the baseline schedule reduces it. They also analyzed the effects of generating initial plans using Branch and Bound methods versus heuristics. Their experiment with buffer sizing techniques indicates that the C and PM method might lead to serious overestimation of the required project buffer size. In addition to the literature reviewed so far the reader is also referred to the following more recent literatures: Herroelen and Leus (2004) recent article provided an extensive review of the baseline scheduling literature and a discussion of procedures for the generation of robust schedules. The article by Elmaghraby *et al.* (2003) provided interesting insights in the use of TOC in resource constrained project scheduling. Liu *et al.* (2006) presented a model of critical chain method considered WIP (Work In Process) for product plan. Tukel *et al.* (2006) introduced two methods for determining feeding buffer sizes in critical chain project scheduling. Both methods integrate project characteristics into the formulation. Specifically, one of them incorporates resource tightness while the other uses network complexity. Rabbani *et al.* (2007) presented a newly developed resource-constrained project scheduling method in stochastic networks by merging the new and traditional resource management methods. Long and Ohsato (2008) developed a fuzzy critical chain method for project scheduling under resource constraints and uncertainty. Kuo *et al.* (2009) explored the due-date performance problem using the concept of the aggregated time buffer in Critical Chain Project Management (CCPM). A simulation model was constructed and the performance of the proposed method is evaluated based on four dispatching rules in a wafer fabrication factory.

In the existing studies, the critical chain schedule is created based on right-justification schedule. While WIP considered, the critical chain method adapts to production plan or project schedule in construction engineering. However, for most projects where time-risk lies in a higher level, all the activities must be executed as early as possible and the critical chain method has no ability to plan and control them. In this study, our goal is to present a new critical chain method which is created based on the active schedule, in which all activities should be executed as early as possible. In most product development projects, such as software development projects, the active critical chain method is convenient to control the risk of project delay.

ACTIVE CRITICAL CHAIN METHOD

According to PMBOK, the critical chain method is a schedule network analysis technique that modifies the project schedule to account for limited resources. Different from CPM and other project management method, where the activity duration is estimated based on 80 or 90% probability of completion, the activity duration in CMM is estimated based on 50% probability of completion. The reduction in time should encourage task performers to start the task on schedule, thereby avoiding the student syndrome. Further, it should also discourage people from deliberately showing their work pace, thereby preventing Parkinson from taking hold. In CCM, it is implied there's a 50% chance that the task will not be completed on time. So, to reassure task estimators/ performers, Goldratt recommends implementing the buffer mechanism: (1) Project Buffers (PB): Amount of buffer time at the end of the project; (2) Feeding Buffers (FB): Amount of buffer time at the end of a sequence of activities which is placed between the chain's end and the merging into the critical chain to protect the critical chain from non-critical chains feeding their delays into it.

We consider a simple project example, as shown in Fig. 1, including start activity 1 and end activity 12, there are 12 activities in the project. Four types of renewable resources A, B, C and D are used, and the availability of each resource is 1 unit per/time period. The resource requirement of all the activities in each time period is also 1 unit. The activity duration, precedence relationships and resource requirements are shown in the network diagram in Fig. 1. A CCM schedule is generated as Fig. 2a.

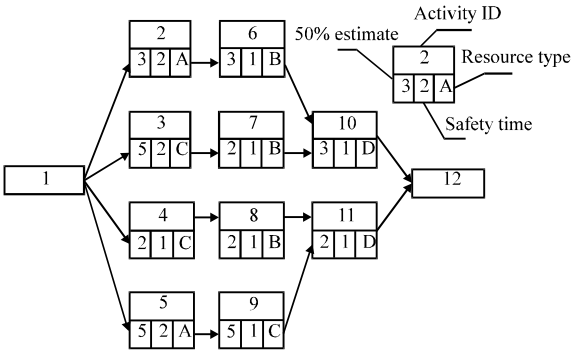

Fig. 1: The illustrative project case

In this study, we address the active critical chain method and an ACCM schedule of the project instance in Fig. 1 is generated as Fig. 2b. Similar to CCM, the activity duration in ACMM is also estimated based on 50% probability of completion, and the critical chain the critical chain is also the sequence of both precedence- and resource-dependent terminal elements that prevents a project from being completed in a shorter time, given finite resources. However, ACCM has two particular characteristics different from CCM as follows:

- All the activities not in critical chain are also executed as early as possible. As shown in Fig. 2b, the non-critical activities 3, 4, 6, 7, 8 and 10 are all started as early as possible to avoid the high risk of time overrun, while the non-critical activities in CCM are started as late as possible to minimize WIP. Accordingly, the ACCM schedule should be generated based on a left-justification schedule while the CCM schedule should be generated based on the right-justification schedule.
- The feeding buffer sizes in ACCM are set as the float time of the non-critical chain directly while they must be computed and inserted to the baseline schedule in CCM. For CCM, according the popular computation method RSEM, the two feeding buffers in this example are both 2, as shown in Fig. 2a. For ACCM, since the baseline schedule is a left-justification schedule, in most case, the float time of non-critical chain is large enough to be regarded as feeding buffer. In Fig. 2b, the two feeding buffers are directly set as the float time of non-critical chain and they are sized 4 and 3, separately.

GENERATION SCHEME OF THE ACTIVE CCM

Here, the schedule procedure of the ACCM will be described and the intermediate steps from a project

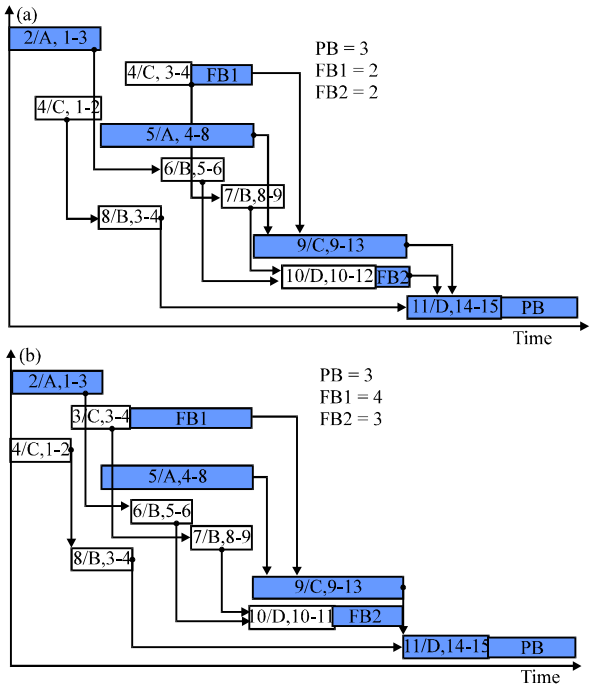

Fig. 2(a-b): Scheduling results of CCM and ACCM, (a) A critical chain schedule and (b) An active critical chain schedule

network to the ACCM schedule will be dedicated in the following subsections by comparing with CCM.

Generate baseline schedule: As mentioned above, the schedule of the baseline of CCM belongs to the resource constrained project scheduling problem (RCPSP). In ACCM, the baseline schedule is an active schedule and thus, all the activities in the baseline schedule are executed as early as possible. For the example as shown in Fig. 1, one of active schedules is shown as Fig. 3a, which will be used as the baseline schedule of the active critical chain schedule. Contrarily, all the non-critical activities in the right-justification schedule are scheduled as late as possible, as shown in Fig. 3b which is used as the baseline schedule of the traditional CCM.

Identify critical chain: According to the definition of critical chain method, the critical chain is series of activities that have zero slack time. In a feasible schedule, there might be several series of activities having zero slack time. Goldratt suggested any one of them can be chosen as the unique critical chain. In this study, we present a new critical activities definition method that the series of activities have the higher priority value should be preferentially chosen as critical activities. The priority values of activities should be predefined, or calculated based on priority rules or other heuristics algorithms, which are the commonly ways in heuristics of RCPSP.

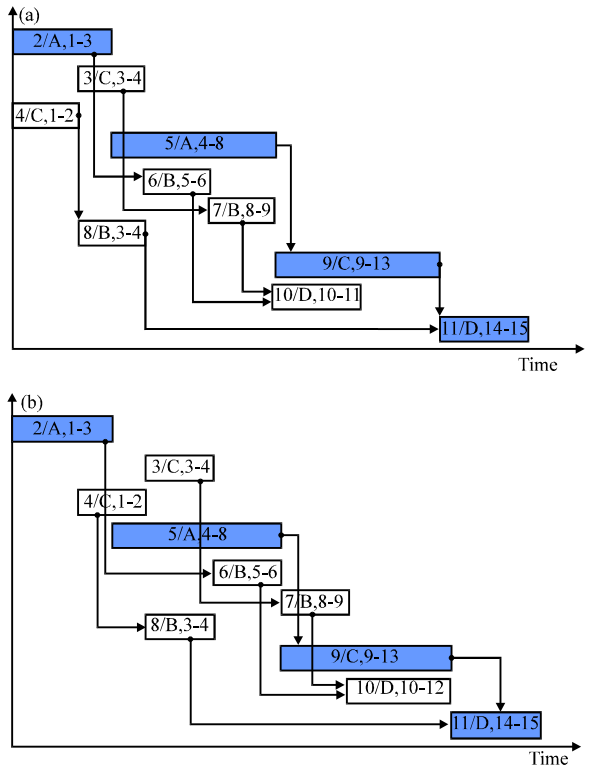

Fig. 3(a-b): Baselines of ACCM and CCM, (a) Baseline of ACCM and (b) Baseline of CCM

Definition 1: In an active schedule, if there are several series of activities that have zero slack time due to precedence relationships or resource constraints, all these activity series are defined as candidate critical chains, all of which may be the unique critical chain of the project.

Definition 2: All the activities in candidate critical chains are defined as candidate critical activities. If a candidate critical chain is chosen as the critical chain, all the activities in it are defined as critical activities.

Lemma 1: In an active schedule, if activity i is a candidate critical activity, all the activities limiting activity i not to be executed in earlier time are candidate critical activities.

Proof: Let:

$$P_i' = \{j \mid f_j = f_i - d_i, j \in P_i\} \quad (1)$$

$$R_i' = \{j \mid f_j = f_i - d_i, r_j = r_i\} \quad (2)$$

P_i is the precedence activity set of i, P_i' and R_i' are two activity sets in which the end time of all the activities is equal to the start time of i. P_i' constrains i not to be started earlier due to the precedence relationships and R_i' constrains i not to be started earlier due to the resource constraints. It should be proved that all activities in P_i' and R_i' are candidate critical activities. Let AC_i be the ordered activity set where all the activities are in candidate critical chain including i but scheduled after activity i, in other words, AC_i is a partial candidate critical chain. Let $j \in P_i' \cup R_i'$, then:

$\because$ $f_j = f_i - d_i$ and AC_i has zero slack
$\therefore$ $\{I\} \cup AC_j$ has zero slack too.

In the candidate critical chain, there must be the other ordered activity set, denoted by BC_j, having zero slack. The reason for it is that if there is no such a set, then activity j can be left moved (the start time is advanced). This is in contradiction with the definition of active schedule. Therefore, $BC_j \cup \{j\} \cup AC_j$ is a candidate critical chain including activity j and it is proofed that $j \in P_i' \cup R_i'$, is a candidate critical activity in the candidate critical chain $BC_j \cup \{j\} \cup AC_j$.

Definition 3: For a candidate critical activity i, all the activities in set $P_i' \cup R_i'$ are defined as precede candidate critical activities.

The end activity of baseline schedule is the end node of critical chain. From the end activity to the start activity, by selecting the activity with the highest priority value in precede candidate critical activities one by one, the unique critical chain can be determined. The algorithm of identifying critical chain from active baseline schedule is formulated as follows:

(1) C: = {n}, lc: = n;
(2) WHILE lc<>1 DO
(3) BEGIN
(4) CC: = $P_{lc}' \cup R_{lc}'$
(5) k←k|v(k) = $\max_{m \in CC}$ {m(j)}
(6) C: = C∪{k}
(7) lc: = k
(8) ENd
(9) LOOP

At first, the algorithm gives an initial critical chain, which includes only the end activity, i.e., C: = {n}. The latest added activity is denoted by lc. At initial stage, lc: = n. In the while loop from 1-9, the candidate critical activity set CC is firstly calculated in (4). Then a candidate critical activity k with the highest priority value in CC is chosen as new critical activity in (5). The new critical activity k is added to the critical chain C in (6).

Subsequently, lc is set as the new critical activity k in (7). Since, the start activity 1 in project instance is a dummy activity without any precedence activities, the executive condition of the while loop defined in (2) is lc <> 1.

For the active schedule shown in Fig. 2a, the critical chain can be identified as {1, 2, 5, 9, 11, 12}, among which activity 1 and activity 12 are dummy activities and not shown there.

Identify non-critical chain: Given a critical activity i, each activity j in $P_i \cup R_i^*$ but not in the critical chain will have a non-critical chain. j is the end activity of the corresponding non-critical chain, which can be regarded as a sub-critical chain ending with j. Therefore, the algorithm of identifying non-critical chain is similar with that of critical chain. The difference of them is that there is no critical activity in non-critical chain. For the active schedule shown in Fig. 2a, there are two non-critical chain: one consists of {4, 3} which connects to the critical activity 9, the other consists of {4, 8, 6, 7, 10} which connects to the critical activity 11.

Compute and insert buffers: The two popular buffer sizing methods are C and PM and RSEM. C and PM sums all of the safety cut from the critical chain (or non-critical chain) and then takes half of this sum and uses it as project buffer (feeding buffer). The obvious advantage of the C and PM is its simplicity. However, the size of the buffer increases linearly with the length of the feeding chain. Compared to C and PM, RSEM has a distinct advantage of not generating very large or very small buffer sizes based on the length of the feeding chain. In this study, RSEM is used to compute project buffer size, as follows:

$$PB = \sqrt{\sum_{j \in C} \delta_j^2} \quad (3)$$

where, PB is the size of project buffer, C is the activity set of the critical chain and σ_j is the standard deviation of the duration of critical activity j. The feeding buffers can also be computed in the same way. According to Eq. 3, the size of project buffer of the illustrative example shown in Fig. 1 can be calculated as:

$$PB = \sqrt{\delta_1^2 + \delta_2^2 + \delta_5^2 + \delta_9^2 + \delta_{11}^2 + \delta_{12}^2} = \sqrt{10} \approx 3$$

In the same way, the sizes of two feeding buffers can be calculated, they are both 2.

Hoel and Taylor (1999) showed that if the overall uncertainty on a given feeding chain is such that the project planner defines a feeding buffer greater than the free slack of the feeding chain, then that chain becomes part of the critical chain. They argue that the feeding buffer sizes can be set as the float time of the non-critical chain, and not be computed by extra method.

In ACCM presented in this study, the active schedule is used as the baseline schedule of the critical chain schedule. That is, in the active baseline schedule, all the activities in the critical chain schedule are started as early as possible under the resource constraints and precedence relationships. As a result, since the float time of the end activity of the non-critical chain is already created and existing in the baseline schedule, all the feeding buffers need not be computed and inserted additionally. For the project instance as shown in Fig. 1, the first feeding buffer can be set as the float time of the non-critical chain {4, 3}, the second can be set as the float time of the non-critical chain {6, 8, 7, 10} and the values of them are 4 and 3, separately, as shown in Fig. 2a.

Although it is not necessary to insert feeding buffers according the buffer sizing method used in this study, the project buffer must be inserted at the end of critical chain. Inserting the project buffer into the baseline schedule, the schedule of ACCM is generated as Fig. 2a.

PROJECT SCHEDULING PROBLEM OF ACCM

Since, ACCM is similar to RCPSP in many ways, we refer the concept model of SMRCPSP (Single-mode RCPSP) to formulate the project scheduling problem of ACCM. The critical chain project network can be represented by an activity-on-the-node network G = (V, E) in which V denotes the set of vertices (nodes) representing the activities and E is the set of edges (arcs) representing the finish-start precedence relationships with zero time-lag. The activities are numbered from 1 to n, where, the dummy activities 1 and n mark the start and the end of the project. All the non-dummy activities are to be performed without preemption. The fixed integer duration of the activity i is denoted by d_i ($1 \le i \le n$), its integer start time is denoted by s_i($1 \le i \le n$) and its integer finishing time is denoted by f_i ($1 \le i \le n$). There are K renewable resource types, each of which, denoted by k, $1 \le k \le K$ has the constant availability denoted by R_k. For an activity i, its constant resource requirement of resource type k is denoted by a_{ik} ($1 \le i \le n$, $1 \le k \le K$).

Different from SMRCPSP, the project scheduling problem of CCM should manage the uncertainty of activity duration, including: (1) the estimated activity duration is shorted 50% of that of RCPSP, (2) the activity series that determines the project duration should be determined as the critical chain (3) the activity series that might cause delay on the start time of critical activities

should be addressed as the non-critical chains (4) the project buffer and feeding buffers should be computed and inserted in the baseline schedule. Since, the feeding buffers need not be considered here, the scheduling problem of ACCM can be formulated as:

$$\text{Min} \quad f_n + PB \tag{4}$$

$$\text{s.t.} \quad f_1 = 0 \tag{5}$$

$$f_i - d_i \geq f_j, \quad j \in P_i \tag{6}$$

$$PB = \sqrt{\sum_{m \in C} \sigma_m^2} \quad \sigma_m = t_m - d_m \tag{7}$$

$$\sum_{j \in A_t} r_{jk} \leq a_k \quad t = 1,2,3.....f_n; \quad k = 1,2,.....K \tag{8}$$

Equation 4 is the objective function minimizing the project make-span, where, f_n represents the project duration before inserting project buffer and PB denotes the project buffer. In Eq. 5, the start time of the dummy start activity 1 is valued 0. The precedence constraints are given by Eq. 6 which indicates that activity j can only be started when all the predecessors are finished. Equation 7 computes the project buffer size by using RSEM. The resource constraints given in Eq. 8 indicate that for each time period [t-1, t] and for each resource type k, the renewable resource amounts required by the activities in progress, denoted by A_t, cannot exceed the availability of k.

THE SOLUTION BASED ON GA

The scheduling problem of ACCM inherits all the characteristics of SMRCPSP, which is a well-known NP hard problem. Since, ACCM should deal with the uncertainty of activity duration additionally, the computational complexity of the scheduling problem of ACCM is higher than that of SMRCPSP and it is very difficult to be solved by using the exact algorithms, such branch and bound or dynamic programming method. The heuristics are popularly alternatives to find the near optimal solution of NP hard problems in acceptable time. The first heuristic methods are those based on priority rules. These methods use a schedule generation scheme, either serial or parallel (for a detailed description of the serial and parallel method (Kolisch and Hartmann, 1999), and one or more priority rules to construct one or more schedules. The other heuristics are the meta-heuristics, which are the latest generation of heuristic algorithms, including Artificial Immune Algorithm (Gao and Liu, 2007), Simulated Annealing (Jayalakshmi and Rajagopalan, 2007), Tabu Search (Mahmood, 2002) and Genetic Algorithms etc (Akhtar, 2007). Among these algorithms, the genetic algorithm is a mature and practical intelligent algorithm and it has been widely applied to solve most kinds of discrete optimization problem (Gao *et al*., 2006; Lo, 2008).

In this study, GA is employed to develop the heuristic solution of the scheduling problem of ACCM. As a widely implemented optimization method, GA is used to transform a population of solutions from one generation to the next in order to lead the search into promising regions of the feasible domain. Since, the project scheduling problem of ACCM is similar to the RCPSP, we refer to the existing algorithm of RCPSP to design the algorithm to solve the project scheduling problem of ACCM. The algorithm has the following features:

- The priority values deduced from activity list are the most important basis to resolve the resource conflicts and determine the unique critical chain
- SSGS is applied to generate active schedule as the baseline schedule of ACCM and SSGS can be extended as ACCM serial schedule generation scheme by adding some critical chain processes, such as searching critical chain, calculating buffer sizes and inserting buffers

Since, the ACCM serial schedule generation scheme is responsible for generating project schedule, the presented genetic algorithm is devoted to determine the priority values encoded in the activity list used to generate the optimal or near-optimal ACCM project schedule by ACCM serial schedule generation scheme.

Chromosome representation: An essential feature of meta-heuristics is the encoding or representation of phenotype solutions into genotype vectors to which the different operators are applied. Several different encoding scheme exist in the heuristics of the project scheduling problem but we limit our presentation to the activity list (permutation-based) encoding given by Alcaraz and Maroto (2001). An activity list is any precedence feasible permutation $\lambda = (j_1, j_2, ..., j_n)$ of the activities. If $I = j_h$, we can say that the activity i is in the position $p(i) = h$. The serial schedule generation scheme transforms an activity list λ into a schedule $S(\lambda)$ by taking the activities one by one in the order of the list and scheduling them at their earliest precedence and resource-feasible start time. This

procedure generates the so-called active schedules, where, each activity is performed as early as possible under the precedence and resource constraints. A schedule S might be presented by several activity lists that only differ in the positions assigned to those activities with the same start times. However, if S is an active schedule and $\lambda(S)$ is any activity list representation of S, there will be $S(\lambda(s)) = S$ if smaller activity number is used as tie-break, the presentation is unique. In our presentation, we assume that the tie-break rule is to choose the activity with the smallest activity label and we denote the unique activity list presentation of S by $\lambda(S)$. The fitness of an individual λ is computed based on the make-span of the associated schedule $S(\lambda)$.

Decoding with ACCM serial schedule generation scheme: Decoding is responsible for transforming the chromosome λ supplied by the genetic algorithm into a complete project schedule. In this study, we present the ACCM serial schedule generation scheme on the basis of the traditional SSGS and the characteristics of ACCM as follows:

- The activity list is imported into SSGS to generate baseline schedule
- Identify critical chain
- Identify non-critical chain
- Compute and Insert buffers

In the critical chain schedule, the make-span of project can be regarded as the value of the objective function.

Fitness function: Based on the make-span of the critical chain project schedule generated previously, the fitness of chromosomes in the population can be evaluated. Let an integer λ denote the chromosome of an individual, $g(\lambda)$ denote the fitness function of λ and $f(\lambda)$ denote the objective function which is the make-span of the schedule. Moreover, f_{max} denote the maximum objective function value and f_{min} denote the minimum objective function value, the fitness value of an individual can be formulated as:

$$g(\lambda) = \frac{f_{max} - f(\lambda) + \gamma}{f_{max} - f_{min} + \gamma} \quad (9)$$

In Eq. 9, $\gamma = 0.1$, which prevents the formulation from being divided by zero.

Selection: Selection is an artificial version of the natural phenomenon called the survival of the fittest. In nature, competition among individuals for scant resources and for mates results in the fittest individuals dominating over weaker ones. Based on their relative quality or rank, individuals receive a number of copies. A fitter individual receives a higher number of offspring and thus it has a higher probability of surviving in the subsequent generation. There are several ways of implementing the section mechanism, such as remainder stochastic sampling without replacement, 2-tournament and ranking. In this paper, we have implemented remainder stochastic sampling without replacement, which has the ability to reduce the stochastic errors associated with roulette wheel selection. In this method, the number of expected copies of each individual N_λ is given by the probability of selecting that individual p_select_λ, multiplied by the population size. Each individual is allocated samples according to the integer part of N_λ and the fractional are treated as probabilities of obtaining another copy. p_select_λ and N_λ are calculated as follows:

$$p\_select_\lambda = \frac{g(\lambda)}{\sum_{j=1}^{POP\_SIZE} g(j)}, \quad N_\lambda = p\_select_\lambda \times POP\_SIZE \quad (10)$$

where, $g(\lambda)$ represents the fitness value of the individual λ.

Two-point crossover: Crossover combines some features of two parent chromosomes to form two offspring which inherit their characteristics of parents. The individuals of the population are mated randomly and each pair undergoes the crossover operation with a probability, producing two children by crossover. The parent population is replaced by the offspring population. For the sequence-based encoding in the activity list, we apply the two-point linear order crossover designed for sequencing problems because no elements are to be repeated in a sequence-based chromosome. Linear order crossover has the characteristic of preserving the relative positions between the genes. For example, two parent chromosomes of five activities, parent 1:(3, -2, 4-, 1, 5) and parent 2:(1, 4, 5, 2, 3), are given with indicated cross sections to be swapped. If plugged into parent 1, the cross section of parent 2 would cause a repetition of activity 5 in the sequence. So, activity 5 is removed from parent 1 and activity 2 is slid into position 4, while activity 1 is slid into position 5. Thus the first child would be (3, 4, 5, 2, 1) and the second child (1, 2, 4, 5, 3) is obtained in the same way.

Mutation: After the crossover operator has been applied and the offspring population has replaced the parent population, the mutation operator is applied to the offspring population. Mutation alters one or more genes of a selected chromosome to reintroduce lost genetic material and introduce some extra variability into the population. In this study, we implemented the mutation operator proposed by Hartmann (2001) in his genetic algorithm. Given a solution, each position is exchanged with the following one (if the result is a feasible solution) with a probability of P_{mut}.

Reproduction mechanism: Some of the best individuals are copied from the current generation into the next. This strategy is called elitist and its main advantage is that the best solution is monotonically improving from one generation to the next. However, it can lead to a rapid population convergence to a local minimum. Nevertheless, this can be overcome by using high mutation rates.

COMPUTATIONAL EXPERIMENTS

Here, we try to adjust the parameters and analyze the computational performance of the genetic algorithm. The experiments are conducted on a PC with Intel Pentium IV 2 GHz CPU and 1024 MB RAM under Windows XP. The program of the presented algorithm is coded in JAVA and compiled with Eclipse 3.0.

Test instances: There are no standard instance sets to evaluate the algorithms of the presented problem. We resort to the single-mode project instance sets in PSPLIB, which are generated using ProGen (Kolisch and Hartmann, 1999). In PSPLIB, there are several project instance sets for single-mode RCSPSP which are J30, J60, J90 and J120, including 30, 60, 90 and 120 activities, respectively. Some modifications are necessary to make these project instances applicable for the scheduling problem of ACCM, as follows:

- Assuming the durations of all the activities in each instance are the average estimates (50%)
- Let d_i denote the average estimate of activity i, ξ be a float valued in [0.1, 1], the standard deviation of d_i is set as $\sigma_i = \xi \times d_i$. If σ_i turns out to be a real number, then it is rounded to the nearest integer

Parameter setting: Some parameters such as the population size (PopSize), iteration generations gen and mutation probability P_{mut} are the parameters influencing the performance of the GA. We use the instance set J30 with 1000 generated schedules to determine the parameter configuration. The predefined parameters are listed in Table 1.

Table 1: Different parameter configuration

Parameters	Values of parameters
PopSize×Gen	{40×25, 25×40, 20×50}
P_{mut}	{0.05, 0.10, 0.15, 0.2, 0.25, 0.30, 0.35, 0.40, 0.45}

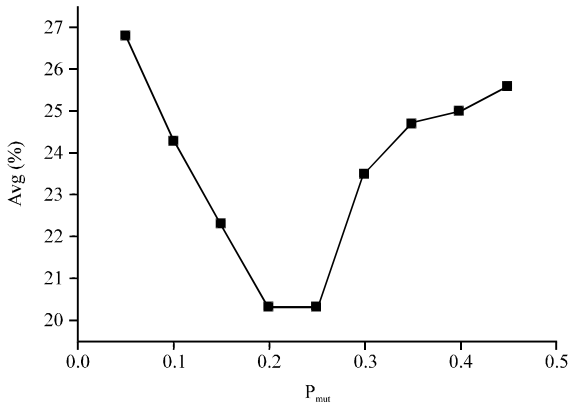

Fig. 4: Impact of the P_{mut} when PopSize = 40, Gen = 25

From the experiment results, we found that different values of PopSize or Gen don't affect the performance of the algorithm when the number of generated schedules is limited. However, different mutation probability P_{mut} will lead to different performances. For comparing the performances of algorithms with different configurations, we determine the lower bound of each project instance firstly. Ignoring the resource constraints, the project is scheduled with traditional CPM (Critical Path Method), the critical path is regarded as critical chain, the project buffer is calculated by using the critical activities sequenced in critical path. Under these assumptions, the make-span of project instance is the project duration calculated by CCM plus project buffer, denoted by L_{val}, which defines the lower bound of the make-span for each project instance. Based on the lower bound, we define the comparison criterion as:

Avg (%): The average percentage of deviation from L_{val}, taken over the set of problems.

Figure 4-6 show the variable performance of algorithms varies with the changes of P_{mut} in different PopSize×Gen.

It can be drawn than the algorithm has the best performance when P_{mut} = 0.2. Based on the experiment results, we value the parameters of the presented GA as follows: PopSize = 40, Gen = 25 and P_{mut} = 0.2.

To further investigate the different performances of presented algorithm for the scheduling problem of ACCM, we test it on a large number of project instances. Figure 7

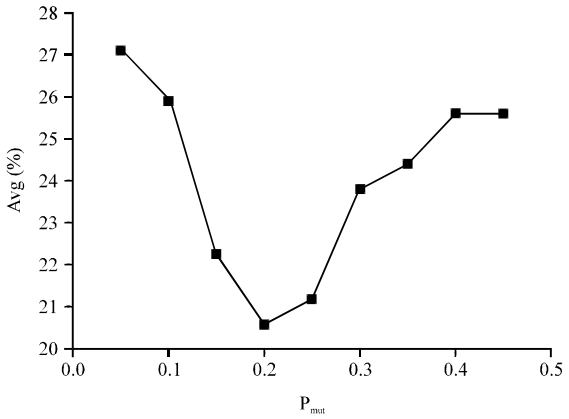

Fig. 5: Impact of the P_{mut} when PopSize = 25, Gen = 40

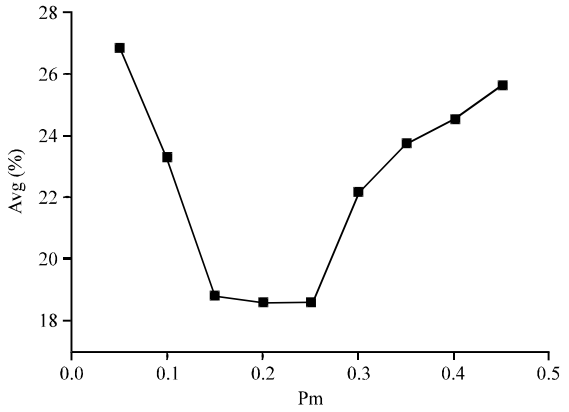

Fig. 6: Impact of the P_{mut} when PopSize = 25, Gen = 40

shows the simulation results of j1201.sm in J120 of PSPLIB, j601_1.sm of J60 and j301_1.sm of J30, it can be shown that the genetic algorithm presented in this study is effective for the scheduling problem of ACCM and it can quickly converge for the project instances with different sizes.

Comparison: There is no any benchmark for the scheduling problem of ACCM. Since, the scheduling problem of ACCM inherits most characteristics of RCPSP, the priority rule-based heuristics can solve the problem in the similar way. Therefore, in this study, we compare the performance of GA with those of heuristics based on priority rules.

The heuristics based on activity priority rules can be divided into two stages: the first stage is computing the priority values of each activity; the second is generating the project schedule with SGS. In most cases, the priority rules and SGS can be implemented separately. Then, the critical chain and non-critical chain can be searched based

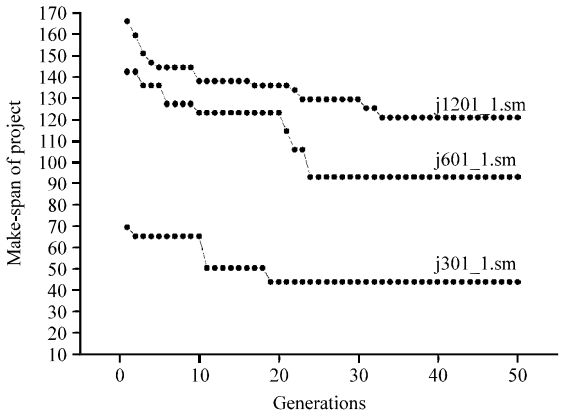

Fig. 7: Calculation examples of the presented algorithm

Table 2: Result comparison of priority rules and presented GA in Avg (%)

	J30	J60	J90	J120
GRD	31.32	29.43	32.56	30.60
LFT	35.44	36.17	36.36	35.85
LST	35.29	36.38	36.17	36.01
SPT	39.78	40.23	41.89	40.58
GRPW	42.67	41.97	42.39	43.56
EST	40.28	41.76	43.72	42.91
MTS	44.54	45.88	46.19	47.02
GA with 1000 schedules	18.21	19.84	19.72	18.73
GA with 2000 schedules	17.75	18.52	18.55	19.18
GA with 3000 schedules	17.72	18.45	18.42	19.23
GA with 5000 schedules	17.68	18.32	18.40	18.79

GRD: Greatest resource demand, LFT: Latest finishing time, LST: Latest starting time, SPT: Shortest processing time, GRPW: Greatest rank positional weight, EST: Earliest starting time and MTS: Most total successors

on the baseline schedule thereby the project buffer is computed and inserted at the end of the baseline schedule and the entire active critical chain project schedule can be achieved.

In RCPSP, priority rules based heuristics have been studied by several authors so far (Kolisch and Hartmann, 1999). When performing the algorithm presented in this study, we only evaluate some popular priority rules used in RCPSP, including Greatest Resource Demand (GRD), Latest Finishing Time (LFT), Latest Starting Time (LST), Shortest Processing Time (SPT), Greatest Rank Positional Weight (GRPW), Earliest Starting Time (EST) and Most Total Successors (MTS).

The results of the comparison are presented in Table 2. The results indicate that when limiting the total number of generated schedules to 1000, our algorithm clearly outperforms all the priority rules, among which the GRD is the most excellent one. It also reveals that our new algorithm performs consistently well over all problem sets. To further investigate the performance of the presented algorithm, we test the presented algorithm with 2000, 3000 and 5000 generated schedules separately. The

computation results listed in Table 2 show when the total number of generated schedules is more than 2000, the 1 performance of the algorithm is hard to be further improved. It can also be concluded that the proposed GA is efficient for the project scheduling problem of ACCM and can be used as a benchmark for future research.

CONCLUSION

In existing CCM, all the non-critical activities are scheduled as late as possible to minimize WIP. However, in quite a few project cases, such as software development project, the minimization of WIP is not always a primary concern, and the project duration is mainly objective. For these projects, to void the risk of project delay, all the activities should be started as soon as possible and then the activities not in critical chain should also be scheduled as early as possible. Therefore, some modifications are necessary to make CCM applicable for these project cases.

We presented a revised critical chain method, defined as active CCM. The core feature of the ACCM is that it uses the active schedule as baseline schedule while the traditional one uses the right-justification schedule. The procedure of generating ACCM schedule is given, some of which are formulated. Furthermore, we propose the project scheduling problem of ACCM and compare the schedule process between ACCM and CCM. The scheduling problem of CCM is a NP hard problem and hard to be solved by exact algorithms, thereby an intelligent algorithm based on GA is investigated to solve it and the effectiveness of the presented algorithm is verified by a full computation experiment.

The ACCM provides an alternative of critical chain method for some project cases in which WIP is not a primary concern and thus it enlarges the application scope of critical chain method. Although the presented GA is similar to the algorithm presented for solving RCPSP, it adds some new features according to the characteristics of the active critical chain method and the achieved computational results can be used as the benchmarks for future research.

ACKNOWLEDGMENTS

This research was funded by National Natural Science Foundation of China under Grant No.71071100 and Science and Technology Project of Liaoning, China under Grant No.2011401010.

REFERENCES

Akhtar, Z., 2007. Genetic load and time prediction technique for dynamic load balancing in grid computing. Inform. Technol. J., 6: 978-986.

Alcaraz, J. and C. Maroto, 2001. A robust genetic algorithm for resource allocation in project scheduling. Ann. Operations Res., 102: 83-109.

Elmaghraby, S.E.E., W.S. Herroelen and R. Leus, 2003. Note on the paper resource-constrained project management using enhanced theory of constraints. Int. J. Pro. Manag., 21: 301-305.

Gao, H. and X. Liu, 2007. Improved artificial immune algorithm and its application on the permutation flow shop sequencing problems. Inform. Technol. J., 6: 929-933.

Gao, H., B. Feng, Y. Hou, B. Guo and L. Zhu, 2006. Adaptive SAGA based on mutative scale chaos optimization strategy. Inform. Technol. J., 5: 524-528.

Goldratt, E.M., 1997. Critical Chain. North River Press, Great Barrington, USA., Pages: 246.

Hartmann, S., 2001. Project scheduling with multiple modes: A genetic algorithm. Ann. Opera. Res., 102: 111-135.

Herroelen, W. and R. Leus, 2001. On the merits and pitfalls of critical chain scheduling. J. Opera. Manag., 19: 559-577.

Herroelen, W.S. R. and Leus, 2004. Robust and reactive project scheduling: A review and classification of procedures. Int. J. Pro. Res., 42: 1599-1620.

Hoel, K. and S.G. Taylor, 1999. Quantifying buffers for project schedules. Pro. Inven. Manag. J., 40: 43-47.

Jayalakshmi, S. and S.P. Rajagopalan, 2007. Modular simulated annealing in classical job shop scheduling. Inform. Technol. J., 6: 222-226.

Kolisch, R. and S. Hartmann, 1999. Heuristic Algorithms for Solving the Resource-Constrained Project Scheduling Problem: Classification and Computational Analysis. In: Project Scheduling: Recent Models, Algorithms and Applications, Weglarz, J., (Ed.). Kluwer Academic Publishers, Berlin, pp: 147-178.

Kuo, T.C., S.H. Chang and S.N. Huang, 2009. Due-date performance improvement using TOC's aggregated time buffer method at a wafer fabrication factory. Exp. Sys. Appli., 36: 1783-1792.

Leach, L.P., 2000. Critical Chain Project Management. Artech House, Boston, Pages: 330.

Liu, S.X., J.H. Song and J.F. Tang, 2006. Critical chain based approach for resource-constrained project scheduling. Acta Automa. Sinica, 32: 60-66.

Lo, C.Y., 2008. Advance of dynamic production-inventory strategy for multiple policies using genetic algorithm. Inform. Technol. J., 7: 647-653.

Long, L.D. and A. Ohsato, 2008. Fuzzy critical chain method for project scheduling under resource constraints and uncertainty. Int. J. Pro. Manag., 26: 688-698.

Mahmood, A., 2002. Scheduling real-time tasks in ditsributed environment with tabu search algorithm. Inform. Technol. J., 1: 153-159.

Nadjafi, B.A. and S. Shadrokh, 2008. An algorithm for the weighted earliness-tardiness unconstrained project scheduling problem. J. Applied Sci., 8: 1651-1659.

Newbold, R.C., 1998. Project Management in the Fast Lane: Applying the Theory of Constraints. St. Lucie Press Boca Raton Pages: 284.

Rabbani, M., S. Baradaran, S.M.T. Fatemi Ghomi and S.S. Hashemin, 2008. Development of a constructive heuristics rule for constrained resource allocation in stochastic networks. J. Applied Sci., 8: 3917-3923.

Rabbani, M., S.M.T. Fatemi Ghomi, F. Jolai and N.S. Lahiji, 2007. A new heuristic for resource constrained project scheduling in stochastic networks using critical chain concept. Eur. J. Opera. Res., 176: 794-808.

Rand, G.K., 2000. Critical chain: The theory of constraints applied to project management. Int. J. Pro. Manag., 18: 173-177.

Shi, Y.J., G.J. Shen and W. Chen, 2011. Solving project scheduling problems using estimation of distribution algorithm with local simplex search. Inform. Technol. J., 10: 1374-1380.

Steyn, H., 2002. Project management applications of the theory of constraints beyond critical chain scheduling. Int. J. Pro. Manag., 20: 75-80.

Tukel, O.I., W.O. Rom and S.D. Eksioglu, 2006. An investigation of buffer sizing techniques in critical chain scheduling. Eur. J. Opera. Res., 172: 401-416.

Wei, C.C., P.H. Liu and Y.C. Tsai, 2002. Resource-constrained project management using enhanced theory of constraint. Int. J. Pro. Manag., 20: 561-567.