Parallel Evolutionary Algorithm in Scheduling Work Packages to Minimize Duration of Software Project Management

Jinghui Hu[1,2], Xu Wang[3], Jian Ren[3⋆], and Chao Liu[3]

[1] AVIC Manufacturing Technology Institute,
Beijing 100024, China
[2] Aeronautical Key Laboratory for Digital Manufacturing Technology,
Beijing 100024, China
[3] State Key Laboratory of Software Development Environment,
School of Computer Science and Engineering,
Beihang University, Beijing 100191, China

Abstract. Software project management problem mainly includes resources allocation and work packages scheduling. This paper presents an approach to Search Based Software Project Management based on parallel implementation of evolutionary algorithm on GPU. We redesigned evolutionary algorithm to cater for the purpose of parallel programming. Our approach aims to parallelize the genetic operators including: crossover, mutation and evaluation in the evolution process to achieve faster execution. To evaluate our approach, we conducted a "proof of concept" empirical study, using data from three real-world software projects. Both sequential and parallel version of a conventional single objective evolutionary algorithm are implemented. The sequential version is based on common programming approach using C++, and the parallel version is based on GPGPU programming approach using CUDA. Results indicate that even a relatively cheap graphic card (GeForce GTX 970) can speed up the optimization process significantly. We believe that deploy parallel evolutionary algorithm based on GPU may fit many applications for other software project management problems, since software projects often have complex inter-related work packages and resources, and are typically characterized by large scale problems which optimization process ought to be accelerated by parallelism.

Keywords: Software project management, Evolutionary algorithm, N-Vidia CUDA, Parallel computing, GPGPU

1 Introduction

In an actual development process of software project, the project manager's responsibility is to properly schedule the tasks of software project development, supervise the programming work of the software engineer and arrange the whole development progress of the software to ensure that the software can be delivered before the deadline [1]. Therefore, the project management problem is not only

⋆ Corresponding author: J. Ren (renjian@buaa.edu.cn)

a very fundamental problem in software engineering, but also a very difficult problem in the actual work for the project manager.

In fact, software project management is an art of staff management and task scheduling in software engineering. It requires an overall understanding of the lifecycle of software development, such as planning tasks, staff organization and so on. Through a survey on the practice of software project management [2–5], we found that the different task management designed by the project manager, such as the arrangement order of the work packages in a project, or the different resource allocation in the same human resources team, will have a great impact on the project's overall duration. Excellent project managers can shorten the overall duration by arranging the order of tasks, making full use of the team's resources or allocating human resources properly. However, at the beginning of a software project, the project managers need to spend a lot of time on the discussion that what kind of difficulties will be faced in the process of development and the detail of resources allocation in software engineering phase [6], such as requirements analysis, system design, system development, system testing etc.

However, an automatic method to properly arrange the entire software project management process is still lacking. We target at two typical difficulties preventing automatic techniques to be realized into a real world project: 1) the complexity of the problem model causes difficulties when a project manager is formalizing the problem he/she is facing, 2) the execution time of the optimization process is quite often unbearable comparing to the result it yields.

In this paper, we introduce a framework to deploy a conventional evolutionary algorithm to solve the work package scheduling problem as a single objective optimization problem. And by catering the most computation intensive portion of algorithm to run on a parallel manner, this framework speeds up the execution of optimization process significantly. With an empirical study on three real world software project data, we show how this framework can ease above mentioned two typical problems that most of the existing optimization techniques to project management are facing. We claim that heuristics provides mangers with intuitive features towards "simple-to-deploy", and parallel computation running on GPU provides faster execution time.

The main purpose of this paper is to push forward the automated software project management powered by the search-based approach, which is an important component of search-based software engineering (SBSE) and also a future development trend of software engineering project management in the era of big data and parallel computing.

The rest of this paper is organized as follows. First, we give a brief background of evolutionary algorithm and software project management problem in Section 2 and 3. Then we introduce the design of algorithm implemented in a parallel manner in Section 4. Section 5 presents the results of empirical study. In Section 6, we provide a brief overview of related work before concluding in Section 7.

2 Evolutionary Algorithm

2.1 Meta-heuristic Algorithms

In the development process of computer science, the design of program algorithm provides much power to promote the advancement of computer science and technology. We know that in solving the similar kind of problem, the computational efficiency of different algorithms is completely different.

Algorithm has become the soul to solve practical problems using computer. In addition to the classical algorithm that can be used to calculate a certain problem at a given time, a class of algorithms is found with different performance. The nature of this algorithm is that no performance guaranteed can be given in theory but is often possible to give the solution of the problem efficiently in practice. This class of algorithms is called meta-heuristic algorithm.

Unlike the traditional algorithm, the meta-heuristic algorithm tries to provide one or more disaggregation in one calculation, and the corresponding optimal solution can be found in the process of searching. The meta-heuristic algorithm can often find a good solution, but there is no way to prove that it will not get a worse solution, or cannot even find solution. The evolutionary algorithm used in this paper usually gives an approximate optimal solution within a given time, but can not to ensure that this solution is best at a given time.

In reality, there are some extreme situations in the process of implementing the heuristic algorithm, that is, some solutions that the meta-heuristic algorithms need are difficult to find or they cannot be found at all. The heuristic algorithm is often used to solve some Non-Deterministic Polynomial complete problems(NPC) because the calculative requirement of NPC is very large so some deterministic algorithms cannot be used to find the optimal solution. However, the meta-heuristic algorithm can get a good answer in a reasonable time when dealing with NPC, so the meta-heuristic algorithm is also a hot field for academic community.

The thought of the meta-heuristic algorithm has been formed for a long time and since its formation, the research on it has not been interrupted. As a result, there are many ways to achieve it. In recent decades, the research of meta-heuristic algorithm has developed rapidly, and there have been many more mature and specific subalgorithms. In the 1970s, Professor Holland [7] proposed genetic algorithm. Genetic algorithm simulates the evolutionary method of biological population to find the optimal solution of some problems, which opens up a new upsurge in research on meta-heuristic algorithm. Ever since the 1980s, a number of new meta-heuristic algorithms emerged [8], such as Simulated Annealing Algorithm, Artificial Neural Network, Tabu Search and etc..

2.2 Evolutionary algorithm

Evolutionary algorithm [9] origins from Darwin's theory of evolution and Mendel's theory of genetic gene. Its essence is an efficient global search algorithm that imitates the biological evolution of nature. In the process of searching for the

optimal solution, it can automatically acquire and accumulate the more optimal individual in the large population, and control search process adaptively. It can converge to the better solution oriented according to the natural law of the strong stronger and the weak weaker. As is known to all, the difficulty of software project management problem is to find the right order of the work package under the precondition of satisfying the constraint of the resource. However, there are numberless constitute of the work package. Traversal is the easiest approach but it requires too much computing resources and results to lower effectiveness. So we need an approach which consumes less time with a perfect solution.

The core of the evolutionary algorithm is the design of encoding of candidate solutions and the fitness function. The evolutionary algorithm first encodes the solutions according to the different requirements of problem. each individual in the population represents a solution to the problem. At the same time, for each solution of the problem, we also need to design the fitness function. The fitness value of different solutions is used to measure the adaptability of different individuals. Individuals with high adaptability are more likely to survive in the evolution process of population. After encoding, there are four steps including initialized population, crossover, mutation and selection. The specific operation of initialized population is to generate the first generation of population randomly; The cross is to product new offspring population by the individual hybridization between parents then the mutation is to mutate certain genes in the individuals and mutated individuals will be involved in population hybridization to maintain the species diversity of the population. As a result, the operation above will bring some novel solutions; Finally, the selection is to choose better individuals in the whole population according to the natural law of winning. In a word, the evolutionary algorithm obtains the optimal solution by the evolution of the population.

3 Project Management Problem

3.1 Project Plan

The core of project management problem is work package scheduling and resource allocation. Figure 1 is a Gantt chart of typical software project management, which illustrates work package arrangement of a real industry software engineering. Generally, all the important phases in software engineering, such as project planning, requirements analysis, design, development, testing, deployment etc., are inseparable from the allocation of work packages. These work packages have a mutually restrictive relationship. Some work packages must be started when other work packages are completed. For instance, the analysis of software requirements is often to take place after the completion of project planning while some work packages can be done at the same time, and there is no impact on each other, such as software development and software testing can often be synchronized. The goal of project manager is using the shortest possible time to complete all work packages within the software project duration.

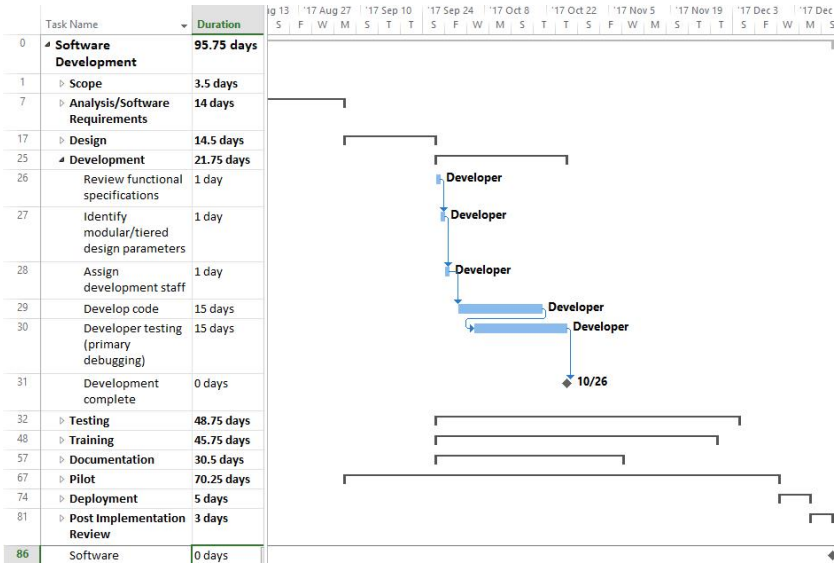


Fig. 1. Software Project Plan in a Microsoft Project File (.mpp)

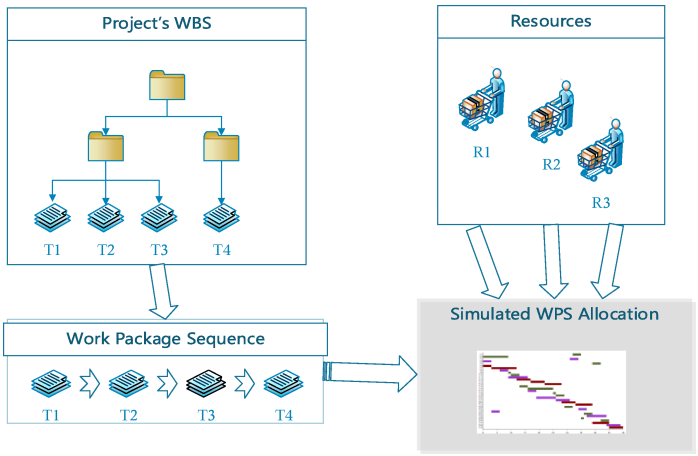


Fig. 2. Project Plan of Software Development

In the process of project management, the work packages in the project are allocated by simulation. See figure 2, Firstly, the whole project is decomposed into several work packages by *work breakdown structure*, and those work packages are arranged into a corresponding *work package sequence*. Secondly, according to the dependencies of the work packages and the restriction of resource, each kinds of resources is allocated to the corresponding work package by *first-come-first-served* algorithm. Finally, the project's overall duration is yielded from the simulation.

3.2 Definition and Assumptions

The goal of this paper is to find an optimal solution or near optimal solution for the project management problem described as follows:

For a software project with N work packages and M kinds of resources, there is a directed acyclic dependencies between these work packages, and each resource can only be assigned to the specified work package. It is necessary to arrange the order of the work package reasonably and make the overall construction duration as short as possible under the condition of satisfying the work package dependency and the resource allocation restriction.

There are three assumptions for our project management problem.

Assumption I: The software project plan can be decomposed into a set of work packages containing N elements $T = \{t_1, t_2, ..., t_N\}$. Each work package in set T is indivisible (i.e., The work package cannot be split to other work package in the schedule), the set of work packages have a pre-estimated workload, the composition of the workload is the collection $E = \{e_1, e_2, ..., e_N\}$. The function $TE : T \to E$, the mean of this function is that for a given work package t_i, produces an estimated workload $e_i = TE(t_i)$ for a work package t_i, where e_i is the estimated workload of t_i.

Assumption II: The work packages in the software project can be processed with M kinds of resources, which constitute a resource set $R = \{r_1, r_2, ..., r_M\}$, for each project's work package set T and the resource set R, There exists a function $TR : T \times R \to \{0, 1\}$, for the given work package t_i and the resource r_j, $TR(t_i, r_j) = 1$ means the resource r_j can be allocated to the work package t_i, while $TR(t_i, r_j) = 0$ means cannot.

Assumption III: All work packages in the set T of the software project plan have dependencies. These dependencies form a set $Dep = \{t_i \to t_j \mid t_i, t_j \in T, t_j \text{ depends on } t_i\}$. As the assumption of this paper, $t_i \to t_j$ means that t_j depends on t_i, that is the work package t_j must be arranged after the work package t_i, and satisfy the formula $t_j.start \leq t_i.end$ (where $t.start$ and $t.end$ respectively indicate the start time and the end time of the work package t). And assume that there is no direct or indirect "loop" dependency between work packages.

3.3 The Objective of Problem

The objective of project management is to find an optimal work package sequence under the three assumptions above. The notation of work package sequence(WPS) is as follows:

$$S = \{(t_{p_1}, r_{q_1}) ... \to (t_{p_j}, r_{q_j}) \to ... (t_{p_N}, r_{q_N}) \mid t_{p_i} \in T, r_{q_j} \in R\} \tag{1}$$

where every (t_p, r_q) means resource r_q is allocated to work package t_p. The WPS needs to meet the following two restrictions:

1. $\nexists i < j, t_j \to t_i \in Dep$. (The WPS must satisfy the dependencies)
2. $\nexists i, k, TR(t_i, r_k) = 0$. (All work packages must have at least one resource)

For the work package arrangement sequence S, each work package t is given $t.start$ (the start time) and $t.end$ (the end time). The total cost duration function of the project is defined as $f(S) = max\{t.end \mid t \in T\}$. That is, the overall duration represents the maximum value of the end time of all work packages in a work package arrangement sequence S, which also means when the last work package of the project is completed, the entire project ends. The objective function is defined as following:

$$\textbf{Objective: } min(f(S)) = min(max\{t.end \mid t \in T\}) \tag{2}$$

The objective of the project management problem is to find a work package sequence S to arrange the whole project under the condition of satisfying the restriction of dependencies and resources, so that the overall duration is minimized.

4 Design of Algorithm

This section introduces the two important steps in using the evolutionary algorithm. One is to choose the representation of the problem's solution, the other is to define a appropriate fitness function.

4.1 The Representation of Solution

For the above-mentioned problem, the representation of solution is defined as follows. This paper uses *work package sequence* (hereinafter referred to as *WPS*) to represent a solution of project management problem. The representation of such a solution is actually a priority arrangement sequence of the work packages in the whole project, and the number of solutions for a project containing the N work packages is $N!$, which is large enough to do random search.

$$T_1 \rightarrow T_5 \rightarrow T_6 \rightarrow T_4 \rightarrow T_8 \rightarrow T_3 \rightarrow T_9 \rightarrow T_2 \rightarrow T_7 \tag{3}$$

For example, equation (3) is solution, the solution represents the work package sequence in the priority of T_1, T_5, T_6, T_4, T_8, T_3, T_9, T_2, T_7, while arranging the work packages. This representation is beneficial and intuitive to programming.

4.2 Fitness Evaluation

The above-mentioned representation of solution make each individual encoded as *WPS*. The fitness value of *WPS* is the project's overall duration, which is calculated by simulating the assignment of work packages in the set of package sequence, see equation (1).

The Algorithm 1 illustrates how to calculate corresponding project overall duration. Firstly, according to first-come-first-served rule, the front work packages in a *WPS* have high priority to get resources, the back work packages has low priority. Secondly, the TR function defines which work package can get which resources, if two work packages that require the same resources, the higher priority one will use the resources firstly, the lower priority one will use the resources

after the higher one releases the resources. Finally, the project's overall duration is the last end time after all work packages are allocated the resources. As we can see from the Algorithm 1, to calculate the overall duration, the algorithm must traverse all N work packages and M kinds of resources, so the average complexity of fitness evolution is $O(N \times M)$.

Algorithm 1 Fitness Evaluation Algorithm

input: WPS, R, TE, TR
output: $duration$
  for i **from** 1 **to** $WPS.length$ **do**
    $t \leftarrow WPS[i]$
    $effort \leftarrow TE(i)$
    for r **in** R **do**
      $r.occupy \leftarrow 0$
    end for
    for r **in** R **do**
      if $TR(t, r) = true$ **then**
        $t.start \leftarrow r.occupy$
        $t.end \leftarrow r.occupy + effort$
        $r.occupy \leftarrow t.end$
        break
      end if
    end for
  end for
  $duration \leftarrow 0$
  for t **in** WPS **do**
    if $t.end > duration$ **then**
      $duration \leftarrow t.end$
    end if
  end for

4.3 Genetic Operators

There are two main types of operators in evolutionary algorithms: *crossover* and *mutation*. **Crossover** In evolutionary algorithms, crossover is a genetic operator used to vary the programming of a chromosome or chromosomes from one generation to the next. In this paper, we use a two-point crossover, that is, two points are selected randomly on the parent organism strings. Everything between the two points is swapped between the parent organisms, rendering two child organisms then exchange corresponding part of chromosomes to get offspring individual. **Mutation** In evolutionary algorithms, Mutation is a genetic operator used to maintain genetic diversity from one generation of a population of genetic algorithm chromosomes to the next. In this paper, we use two-point exchange mutation, that is, two points is selected randomly on individual organisms strings, and then exchange the two points.

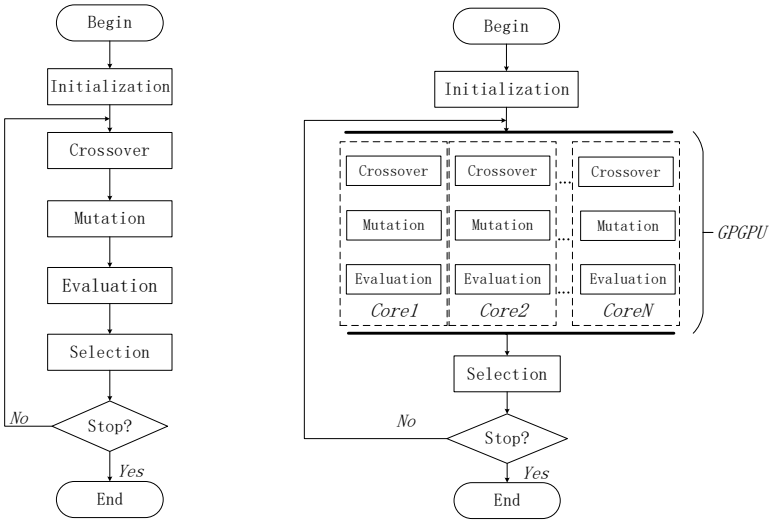
4.4 Parallel Evolutionary Algorithm

Figure 3 summarizes the basic process of the evolutionary algorithm. Figure 3(a) is common sequential evolutionary algorithm, the algorithm has several steps, such as initialization, crossover, mutation, and selection. Candidate solutions to the optimization problem play the role of individuals in a population, and the fitness function determines the convergence of the solutions. Evolution of the population then takes place after the repeated application of the above operators. Figure 3(b) illustrates a parallel evolutionary algorithm. The parallel one follows most steps in sequential one, but put the top time-consuming work on GPU. The steps of crossover, mutation and evolution runs in different cores in GPU so that the speed of calculation can be improved a lot.

4.5 Runtime Environment

The hardware environment of algorithm implementation is as follows: the runtime processor that executes the sequential evolutionary algorithm is the Intel i7 series CPU (abbreviated as CPU). The runtime processor of the parallel evolutionary algorithm is the NVidia GeForce GTX 970 (abbreviated as GPU). The major difference between CPU and GPU is that they have different number of computation cores. Normally, the computation speed of a GPU core is lower than a CPU core. But GPU has far more than the number of CPU cores. In this paper, The CPU we used has 8 cores and the GPU has 1664 cores.

The software environment that runs application is as follows: the sequential programs runs on Microsoft Visual Studio 2012 C++ Compiler environment, the parallel one runs on CUDA 7.0.28 Runtime environment. Both sequential and parallel application is evaluated in the same computer.



(a) Sequential (b) Parallel

Fig. 3. Sequential and Parallel Evolutionary Algorithm

5 Experimental Results

To assess the evolutionary algorithm proposed in this paper, two separated evaluation experiments are made for the effectiveness of evolutionary algorithm and the efficiency of parallel evolutionary algorithm in project management problem. The purpose of the experiment is to address the following two research questions:

RQ1: Does the evolutionary algorithm effectively optimize project management problem, and get an optimized solution?

RQ2: Is the parallel evolutionary algorithm able to improve the efficiency in the project management problem?

5.1 Industrial Project Data

Three industrial project plans have been used in the experimental studies for this paper, which are named as *A-Input*, *B-DBUpgrade* and *C-SmartPrice*. These data come from real world projects and have fundamental details, such as work packages, resources and dependencies. Table 1 shows the number of each project.

Table 1. Three Project's Parameters

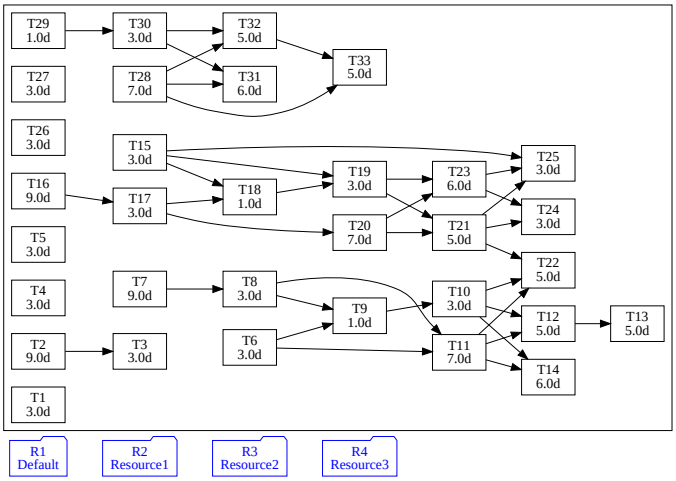
	# work packages	# resources	# dependencies
A-Input	33	4	33
B-DBUpgrade	106	8	105
C-SmartPrice	74	14	73

A-Input is a simulated small-scale project planning. it is a team work plan for finishing class assignments. *B-DBUpgrade* is a real software development plan, which's goal is to upgrade the Oracle database from the *9g* version to the *10g* version. *B-DBUpgrade* is Oracle's non-public version of the project planning. There were different layers of the organisation involved including DBAs, BSAs, developers and users. Furthermore, the project also included the training of the staff for the new features of the system. *C-SmartPrice* consists of a supply chain enhancement of medium size affecting mostly the website as well as a few internal applications [10].

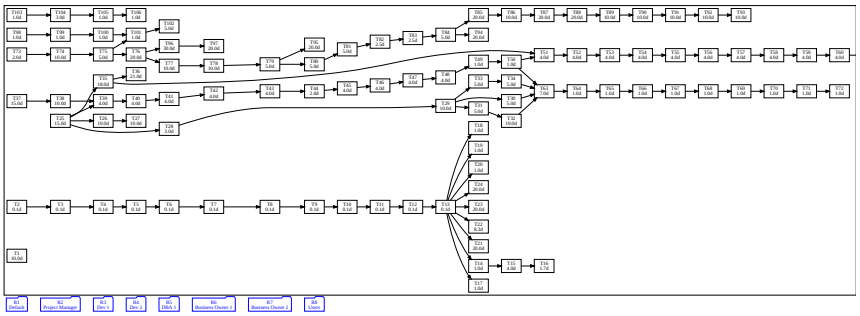
Figure 4 illustrates the relationship between work packages and resources. In each sub-figure, the rectangle represents a work package in the project plan, the content in the rectangle is the work packages' ID and duration; the arrows between work packages suggest dependencies, notice that some redundant dependencies is removed; the folders in the bottom are some certain types of resources that can be allocated to work packages.

5.2 Effectiveness Experiment for Algorithm

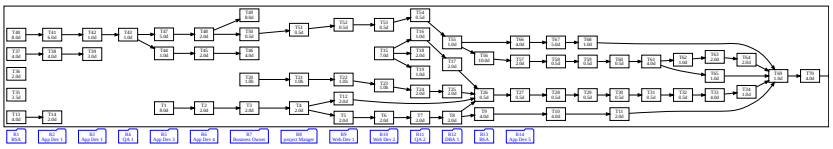
In order to answer **RQ1**, this section designs an effectiveness experiment for evolutionary algorithm. The basic configuration used in the experiment is that 100 individuals runs in 50 generations. For each generation in the evolutionary algorithm, we evaluate each individuals' fitness value in the whole population, and then plot the statistical data into a boxplot diagram.



(a) *A-Input*, 33 work packages, 4 resources and 33 dependencies



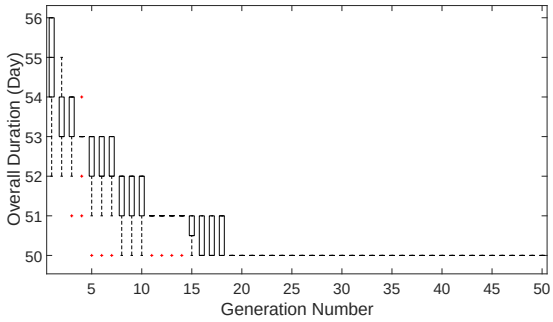
(b) *B-DBUpgrade*: 106 work packages, 8 resources and 105 dependencies



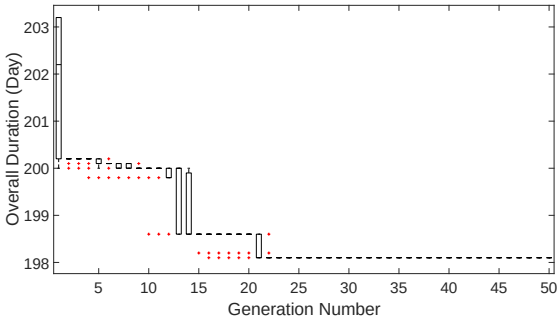
(c) *C-SmartPrice:*, 74 work packages, 14 resources and 73 dependencies

Fig. 4. Three Projects' *work packages*, *resources* and *dependencies*

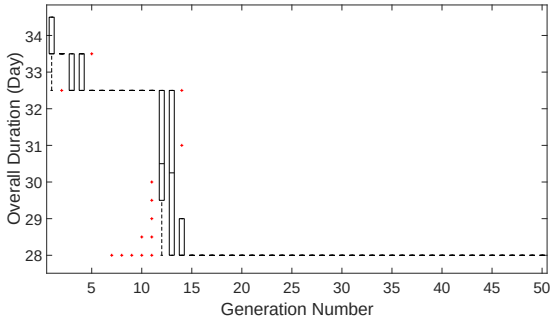
The boxplot diagram shows the five kinds of statistical data of all individuals' fitness value in each generation. The data are the minimum, the lower quartile, the median, the upper quartile and the maximum values of the whole population. Figure 5 shows the results of the above-mentioned three industrial projects. In each sub-figure, the tick labels on the horizontal axis indicate the total number of internal generations that have been carried out, and also indicates the point at which the algorithm updated the population used for fitness computation. At each point on the horizontal axis, the entire population is depicted using a boxplot to give both a sense of the values obtained for completion time as the evolution progresses and the distribution of the fitness values in the population.



(a) *A-Input*'s solution converges on *19th* generation



(b) *B-DBUpgrade*'s solution converges on *22rd* generation



(c) *C-SmartPrice*'s solution converges on *15th* generation

Fig. 5. Statistics of Fitness Value in Each Generation

From the trend of three projects in the figures, it can be seen that the population is diversified when the population is initialized and as the number of generation increase, all the fitness value begins to decrease, and finally converges to an optimized result.

The convergence of solutions is quite fast and stable. Figure 5(a) shows the solution of *A-Input* converges on 19th generation. Figure 5(b) shows the solution of *B-DBUpgrade* converges on 22rd generation. Figure 5(c) shows the solution of

C-SmartPrice converges on 15th generation. Those results show us evolutionary algorithm will find an optimized overall duration for specific project plan.

Through the above analysis results of experimental data, we can accurately answer **RQ1**. In the project management problems, the evolutionary algorithm does have a good optimization and improvement.

5.3 Efficiency Experiment for Algorithm

In order to answer **RQ2**, this section conducts an efficiency experiment that compares the efficiency between the parallel evolutionary algorithm and the sequential one.

Table 2. The Comparison of Sequential and Parallel Implementation

	A-Input		*B-DBUpgrade*		*C-SmartPrice*	
Time(ms)	CPU	GPU	CPU	GPU	CPU	GPU
1	7991.8	4102.5	45225.1	21217.5	29642.9	14844.3
2	8431.8	4279.7	44832.1	21377.9	28733.7	14543.6
3	7504.1	3947.9	44934.9	21227.2	28657.9	15003.4
4	7442.1	4642.5	44197.1	21340.3	29489.5	14882.8
5	7376.9	4263.8	45233.7	22448.2	29379.5	14921.5
6	7385.4	4188.6	45197.5	20956.1	28687.9	14723.3
7	8238.8	4540.2	45305.4	21777.2	29786.3	13978.8
8	7470.7	4226.4	45322.8	21147.1	31231.7	14811.1
9	7604.8	3778.2	45040.7	21004.7	28319.7	14292.0
10	7451.9	4712.4	45970.6	21500.9	28322.1	14429.7
Avg.	7689.8	4268.2	45126.0	21399.7	29225.1	14643.1

The initial configuration of the two algorithms is the identical, which the number of population is 1000 and the number of generation is 100. Each generation runs several steps including crossover, mutation and selection etc. A timer is set to record the total executing time of the experiments. The timer starts after loading initial data and ends after the result is calculated. This operation allows us to count the real executing time both in the sequential environment and the parallel environment under the same criteria. The efficiency experiments are repeated 10 times. Table 2 shows the comparison of sequential and parallel implementation executing time on all three industrial projects. By the comparison of average executing time of 10 times results, the average executing time of *A-Input* on CPU is 1.80 times than GPU, the average executing time of *B-DBUpgrade* on CPU is 2.10 times than GPU and the average executing time of *C-SmartPrice* on CPU is 2.00 times than GPU.

In summary, the executing time of sequential algorithm on CPU is roughly twice as long as the parallel algorithm on GPU. So it can be a good answer to **RQ2** that parallel evolutionary algorithm can improve the efficiency on computing the project management problems.

6 Related Works

In 1993, Chang et al. [2] first proposed the project management problem. The view of Chang is that software project management net(SPM-Net) can be used

to schedule tasks and manage the resources of software development. In his article, Chang's project management problem is based on the simulative data, the reason leading to this is the real industrial data of software project management is very scanty. In 2007, Alba and Chicano [3] optimized the search algorithms for project management, and solve the project management problem using genetic algorithm. Their goal is using a search-based approach to reduce the final completion duration of a project. In 2009, Ren et al. [4] first applied co-evolutionary algorithms to solve project management problem. Recently, Sarro et al. [11] proposed a a multi-objective decision support approach to help balance project risks and duration against overtime, so that software engineers can better plan overtime. At present, the search space of the work package based on the project management is more and more huge, the sequential algorithm is not so effective to solve such problems. Thus, finding a parallel algorithm has become a hot topic on research [6].

In recent years, search-based project management problem has become an important branch of search-based software engineering, and has become a new field of research. At the same time, the number of papers related to search- based project management problem is also rising, which makes many researchers willing to engage in search-based project management problem, so in turn provides a new platform for practice and innovation of the search-based project management problem [5]. In 2011, Thomas [12] proposed an approach to describe software project problems with a set of multi-objective criteria for portfolio managers using the COCOMO II model and introduce a multi-objective evolutionary approach. In 2014, Samanta [13] gone through a very brief idea on Genetic Algorithm and implemented the algorithm using MATLAB. The search-based algorithm is a compute-intensive method, which means the computer's CPU will be used usually consumed a lot. Therefore, the traditional sequential computing model cannot meet the requirement of increasing calculation speed. In 2007, Alba and Chicano [14] began using search-based methods to improve the optimal solution of problems, and for the first time using a parallel code model to test the efficiency of search-based methods. In 2013, Jing [15] began solving software project scheduling problems with ant colony optimization. In recent years, more search-based software engineering methods (such as simulated annealing, climbing algorithms, evolutionary algorithms, tabu search, etc.) have been used to solve project management problem, and these methods are usually able to get good convergence solution on project management problem.

As the development of the algorithm, how to speed up the calculation of the algorithm has become a hot spot in academic community. In 2012, Zhang et al. [16] apply GPGPU to simulation for complex scenes and later on they proposed a GPU-based parallel MOEAs [17]. CUDA is a generic parallel computing architecture developed by NVIDIA, which can be used to improve the performance of GPGPU [18] and also can be optimized by meta-heuristic algorithms [19, 20]. In 2014, Jianmei [21] proposed five novel parallel algorithms for solving multi-objective combinatorial optimization problems exactly and efficient. Compared to our work, those algorithms are different in solved questions,

but there is the same thought of parallelization to gain the faster speed. They apply the algorithm to the case of software-system designs and get great solution, which illustrate the great characteristic of parallelization in the domain of software engineering. In general, the parallel evolution algorithm can be implemented on the GPU platform using the CUDA [22]. With the development of the technology of parallelism, a general-purpose GPU(GPGPU) is used to improve the speed of calculation in software engineering.

Furthermore, a common project management tool which deal with the mathematical models have not been implemented, so it is difficult to apply the theory to the industrial project management process. In 2012, Stylianou [23] and Gerasimou first developed a tool for project management, which they named IntelliSPM. The tool uses Matlab and Java programming language and supports staffing arrangement and resource allocation optimization. In their work, Stylianou uses the fitness function to dynamic calculate the dependencies between work packages, so the real project's dependencies may be broken during the calculation. So in current software engineering practice, the tool supporting project management is still lacking.

7 Conclusions and Future Work

This paper introduces a framework to solve the work package scheduling problem as a single objective optimization problem. The framework accelerates the optimization process by parallelizing the evolving of solutions on a graphic card. We conducted a "proof of concept" empirical study using data from three industrial software projects, aimed at demonstrate the effectiveness and efficiency of proposed framework. Results show that the population converge around 20 generations and the parallel version of single objective evolutionary algorithm spend half of the execution time compared to the sequential version, even with only one relative cheap hardware. Therefore, we claim that heuristics provides manager with intuitive features towards "simple-to-deploy", and parallel computation running on GPU provides shorter execution time.

Future work aims at extending the study reported in this paper with further data sets and accelerating hardware with more cores, above all, at considering a more sophisticated project model, which accounts for further factors not considered in this study, such as group configuration of staff and the interaction among staffing and scheduling. As the development of the scale of the software project, there are more than one objectives in the problem define. so we will increase the complexity of the model of the software project management problem and improved our algorithm to solve multi-objective problem. Future work will also include developing a set of automation tools and techniques. We believe that a optimization tool can automatically parsing a Microsoft Project file (.mpp) can be made useful to provide optimized solutions to aid the managers in practice.

Acknowledgements: This work was supported by the National Natural Science Foundation of China (61602021) and the State Key Laboratory of Software Development Environment (SKLSDE-2017ZX-20).

References

[1] Stellman, A., Greene, J.: Applied Software Project Management. OREILLY, (2005)
[2] Chang, C. K., Jiang, H., Di, Y., Zhu, D., Ge. Y.: Time-line Based Model for Software Project Scheduling with Genetic Algorithms. Info. & Softw. Tech., 1142-1154 (2008)
[3] Alba, E., Francisco, C. J.: Software Project Management with GA. Information Sciences, 2380-2401 (2007)
[4] Ren, J., Harman, M., Penta, M. D.: Cooperative Co-evolutionary Optimization of Software Project Staff Assignments and Job Scheduling. GECCO, 495-519 (2011)
[5] Penta, M. D., Harman, M., Antoniol, G.: The Use of Search-based Optimization Techniques to Schedule and Staff Software Projects: An Approach and an Empirical Study. Software Practice and Experience, 495-519 (2011)
[6] Pentico, D. W.: Assignment Problems: A Golden Anniversary Survey. European Journal of Operational Research, 774-793 (2007)
[7] Holland, J. H.: Adaptation in Natural and Artificial Systems: an Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence. U Michigan Press, (1975)
[8] Harman, M., McMinn, P., Souza, J. T. D., Yoo, S.: Search Based Software Engineering: Techniques, Taxonomy, Tutorial. Empirical softw. eng. and ver., 1-59 (2012)
[9] Deb, K., Pratap, A., Agarwal, S., Meyarivan, T.: A Fast and Elitist Multiobjective Genetic Algorithm: NSGA-II. IEEE TEC, 182-197 (2002)
[10] Ren, J.: Search Based Software Project Management. University of London., 1-178 (2013)
[11] Sarro, F., Ferrucci, F., Harman, M., Manna, A., Ren, J.: Adaptive Multi-objective Evolutionary Algorithms for Overtime Planning in Softw. Projects. TSE, (99) (2017)
[12] Thomas K., Jiri K., Stefan B.: Software project portfolio optimization with advanced multiobjective evolutionary algorithms. Appl. Soft Comput. 11(1): 1416-1426 (2011)
[13] Samanta, S.: Genetic Algorithm: An Approach for Optimization (Using MATLAB). International Journal of Latest Trends in Eng. and Tech., 261-267 (2014)
[14] Pospichal, Jaros, J., Schwarz, J.: Schwarz. Parallel Genetic Algorithm on the CUDA Architecture. Lecture Notes Computation Science, 442-451 (2010)
[15] Jing X., Xian-Ting A., Yong T.: Solving software project scheduling problems with ant colony optimization. Computers & OR 40(1): 33-46 (2013)
[16] Zhang, F., Li, Z., Wang, B., Xiang, M., Hong, W.: Hybrid general-purpose computation on GPU (GPGPU) and computer graphics synthetic aperture radar simulation for complex scenes. International Journal of Physical Sciences, 1224-1234 (2012)
[17] Li, Z., Bian, Y., Zhao, R., Cheng, J.: A Fine-Grained Parallel Multi-objective Test Case Prioritization on GPU. SSBSE'13, 111-125 (2013)
[18] Langdon, W. B., Lam, B. Y. H., Petke, J., Harman, M.: Improving CUDA DNA Analysis Software with Genetic Programming. GECCO'15, 1063-1070 (2015)
[19] Langdon, W. B., Lam, B. H. Y., Modat, M., Petke, J., Harman, M.: Genetic improvement of GPU software. GPEM, 18(1):5-44 (2017)
[20] Langdon, W. B., Harman, M.: Optimizing Existing Software With Genetic Programming. IEEE Trans. Evol. Comput., 19(1),118-135, (2015)
[21] Jianmei G., Edward Z., Rafael O., Derek R., Krzysztof C., Sven A., Joanne M. A.: Scaling exact multi-objective combinatorial optimization by parallelization. ASE 2014: 409-420 (2014)
[22] Vidal, P., Alba, E.: A Multi-GPU Implementation of a Cellular Genetic Algorithm. 2010 IEEE World Congress on Computational Intelligence, 1-7 (2010)

[23] Stylianou, C., Gerasimou, S., Andreou, A. S.: A Novel Prototype Tool for Intelligent Software Project Scheduling and Staffing Enhanced with Personality Factors. International Conference on Tools with Artificial Intelligence. 277-284 (2012)