并行演化算法在项目管理问题上的设计与实现

[1]，[1]，***[1]
北京航空航天大学计算机学院，北京市 100191

摘　　要：本文专注于软件项目经理的对软件项目的计划的需求。首先我们简要介绍了软件项目管理问题（SPMP）的背景和现状。软件项目管理的问题主要包括资源配置和工作包调度。我们的目标是尽量减少软件项目的总工期，同时满足软件项目的工作包之间的依赖关系和资源配置限制约束。求解上述的软件项目管理问题是 NP 难。我们借鉴了基于搜索的软件工程中方法来分析和解决软件项目管理的问题。我们同时实现串行和并行两个版本的应用，这些应用的目的是解决软件项目管理问题。串行版本的应用是使用 C++编程语言基于通常的编程方法实现的，并行版本的应用是使用 CUDA C++接口基于 GPGPU 编程的方法。为了满足我们在 GPU 上编程的意图，我们重新设计了基于搜索的演化算法。最后，我们做了对比实验，验证了并行演化算法确实提高计算效率和演化算法总是能收敛到（近似）最优解。

关　键　词：软件项目管理；演化算法；并行优化算法
中图分类号：TP391
文献标识码：A　　**文章编号：**2017060706

Solving Project Management Problem with Paralleled Evolutionary Algorithm

***[1], ***[1], ***[1]

School of Computer Science and Engineering, Beihang University, Beijing 100191

Abstract: In this paper, we focus on software project managers' needs for software project planning. Firstly, we briefly introduce the background and current state of Software Project Management Problem (SPMP). The software project management problem mainly includes resources allocation and work packages scheduling. Our goal is to minimize the overall duration of a software project, while satisfying the dependencies between work packages and constraints of resources allocation in the software project. Finding an optimal solution for above-mentioned software project problem is NP-hard. We learn from search based software engineering approach to analyze and solve software project management problem. We implement both sequential and parallel version applications, which are aim to solve the software project management problem. The sequential version application is based on common programming approach using C++ programming language, and the parallel version application is based on GPGPU programming approach using CUDA C++ API. We redesign search based evolutionary algorithm to cater for our purpose of parallel programming on GPU. Finally, we conduct a comparison experiment to verify the parallel evolutionary algorithm does improve computational efficiency and evolutionary algorithm always converge to (nearly) optimal solutions.

Key words: Software project management, Evolutionary algorithm, Paralleled Optimization Problem

引言

在生活生产中，我们每个人都会接触到形形色色地生产任务，对于这些大大小小的生产任务，如何正确合理地调配任务是一个亟待解决的热门问题。例如，在一个汽车生产的流水线上，生产经理需要将生产一辆汽车的项目拆分成生产汽车零件或者更加精细的工作包，然后将这些工作包交付给每一台承载加工任务的机器上。不仅在汽车生产过程如此，软件项目管理也是依法炮制。我们知道，在一个实际的软件项目开发过程中，项目经理的职责就是合理调度软件项目开发任务，监督软件工程师的编程工作，同时安排软件的开发执行进度，确保软件能在预定的工期到来之前交付软件项目[1]。因此，项目管理问题是软件工程中的一个非常基本的问题，也是项目经理在实际工作中最为棘手的问题。

通过实际的生产实践调研，我们可以得出，项目管理者设计的不同的任务管理方案，例如在一个项目中的不同的工作包的安排次序，或者同等的人力资源的团队中不同资源分配情况以及安排团队的成员的合理性，这些因素都对一个项目的完成整体进度造成很大的影响。优秀的项目管理者可以通过的合理调配任务的执行次序来缩短一个软件项目的总工期，或者充分利用生产团队的资源，合理分配人力资源，从而最大化地发挥一个团队的整体实力，来为一个项目争取到最大化的收益。但是，公司的项目管理人员在启动一个软件项目之前，往往需要花费大量的时间来研讨软件项目的开发过程中将会遇到什么样的困难，同时要多次研讨才能制定软件项目执行过程中的需求分析，系统设计，系统开发以及系统测试等等相关工作包的分配方案。工业界面临着缺少自动化的方法来合理安排整个项目的工作包，这不仅是项目经理在实际工作过程中所面临的比较难以解决的工作问题，也是工业界要集中解决的挑战。因此，设计出一个能够帮助项目经理管理工作包的工具是非常有必要的。

软件项目管理（Software Project Management）是软件工程中人员管理和任务调度的一门艺术。它需要对软件开发的生命周期，例如计划任务，组织人员等等的各个方面有一个总体的了解。本文主要研究点是使用基于搜索的方法来解决软件项目管理问题相关问题，这也是基于搜索的软件工程（Search Based Software Engineering，SBSE）的一个重要的组成分支，同时也是在大数据时代下软件工程项目管理的未来发展趋势。

1 相关研究工作

1993 年，Chang, C.在相关文献中提出项目管理问题[2]。Chang 的观点是使用软件项目管理网（Software Project Management Net，SPM-Net）技术来调度任务和管理软件开发资源。文献中，Chang 研究的项目管理问题是基于电脑模拟的工业相关数据，导致这一情况的原因是可以利用的真实世界上的软件项目工业数据十分匮乏。随后在 2002 年，Aguilar-Ruiz 等使用模拟的软件项目的数据进一步研究，提出了基于搜索的项目管理方法，他们的方法是提出使用模拟安排工作包的方法来给项目管理者提供可以参照的安排任务的方案[3]。和 Chang 一样的是，Aguilar-Ruiz 也是基于模拟数据来评估他们的实验效果。2007 年，Alba 和 Chicano 对项目管理相应的搜索算法进行优化，提出了使用遗传算法（Genetic Algorithm）来解决项目管理问题，使用基于搜索的方法来定向地降低一个项目的最终完成的总工期。2009 年，Ren 提出了使用协调演化算法的方式来解决项目管理问题[4]。目前，基于项目管理的工作包的求解搜索空间越来越庞大，传统对这类问题的串行的解决方法已经不能满足人们对计算速度的要求了。于是，这类问题的并行方法的解决方案已经成为研究的热门[5]。

近些年来，基于搜索的项目管理问题作为基于搜索的软件工程的一个重要组成分支，已经成为学者开始从事的研究新领域。同时，基于搜索的项目管理问题相关的论文数量也在不断地攀升，这使得很多研究人员愿意从事基于搜索的项目管理问题的研究，也为基于搜索的项目管理问题这一研究方向提供了可以创新和实践的新

平台[6]。

由于基于搜索方法是一种计算密集的方法，通常十分消耗计算机的 CPU 计算资源，因此，传统的串行计算模式已经无法满足基于搜索方法对计算速度的要求。2007 年，Alba 和 Chicano 开始使用基于搜索的方法来提高寻找问题的最优解，同时第一次使用了并行化的代码实现方式来检验使用基于搜索的方法的效率[7]。近几年，更多中的基于搜索的软件工程方法（如模拟退火，爬山算法，演化算法，禁忌搜索等）被用于求解项目管理问题，通常这些方法都能够在项目管理问题上得到不错的收敛解[8]。

目前，针对项目管理问题的定义和求解这一问题，通常学者是在研究的过程中建立的相关数学模型，然后在自己建立的模型上进行研究和优化。这些数学模型通常没有被开发成为一个通用的工具，因此很难恰当地运用到项目经理的实际生产工作过程中。2012 年，Stylianou 和 Gerasimou 第一次开发出来可以实际进行项目管理的工具，他们将这个工具命名为 IntelliSPM[9]。这是一个使用 Matlab 和 Java 编程实现的可以支持软件项目管理的工具，他支持人员安排和资源分配的优化。Stylianou 通过适值评估函数来控制工作包之间的依赖关系，所以这往往不能完全解决工作包的依赖。Stylianou 没有将这个工具公开，目前，在工程实践中，项目管理依然缺乏工具支持。面对着日益攀升的软件项目经理对自动化安排任务工具的需求，开发出一个支持项目经理决策的软件项目管理工具已经成为了基于搜索项目管理领域的新目标。

2 演化算法介绍

2.1 元启发式算法

计算机科学发展过程中，程序算法设计提供了许多推进计算机科技前进的动力。我们知道，在解决同样的一类问题上，不同算法的计算效率是完全不一样的，算法已经成为计算机解决实际问题的灵魂。除了目前研究的可以在指定时间确定地计算出一个问题的算法外，人们还发现一类算法，目前无法在理论上给出任何性能保证，但在实践中往往可以高效地给出问题的解。这样的算法被称为元启发式算法（Meta-heuristic Algorithm）。

与通常固定的算法套路不同的是，元启发式算法试图在一次计算中提供一个或多个目标，在运行中可以找到相应的较优解。元启发式算法常能发现很不错的解，但也没办法证明它不会得到较坏的解，甚至于有的时候根本无法发现解。本文使用的演化算法通常可以在一个预订时间内给出一个近似最优的答案，但通常没有办法保证在给定时间内一定能够给出一个足够优的解。

现实中，实现元启发式算法的过程中也会遇到一些极端的情况，就是一些元启发式算法要寻求的解很难出现，或者根本无法出现。元启发式算法常用来解决一些 NP 完全问题，因为理论上的 NP 完全问题的计算量比较大，无法使用一些确定性的算法来求出较优解。但是，元启发式算法在处理这类的 NP 完全问题时，通常可以在合理时间内得到不错的答案，因此元启发式算法也是学术界研究的一个热门领域。关于元启发式算法的思想形成由来已久，自它形成以来，人们对其的研究就没有中断过，目前也有很多实现的方法手段[10]。近几十年来，元启发式算法的研究发展的很快，先后出现了许多比较成熟具体的算法。上世纪 70 年代，Holland 教授的提出了遗传算法[11]（Genetic Algorithm）。遗传算法模拟生物学种群进化的方法来寻找一些问题的最优解，它开启了人们对元启发式算法的新一轮研究。80 年代，出现了一批新生的元启发式算法，例如模拟退火算法（Simulated Annealing Algorithm），人工神经网络（Artificial Neural Network），禁忌搜索（Tabu Search）。现在，比较热的元启发式算法有演化算法（Evolutionary Algorithm），蚁群算法（Ant Algorithms），遗传量子算法（Genetic Quantum Algorithm）。元启发式算法的研究一直在想着一个新的高度发展[12]。

2.2 演化算法

演化算法借鉴了达尔文的进化理论以及孟德尔的基因遗传学说。它的本质是模仿自然界生物演化的一种高效的全局搜索的算法，它能在自动搜索较优解的过程中自动获取和积累庞大的搜索空间的较优的知识，并自适应的控制搜索过程，根据优者更优，强者越强的自然法则来定向地以求得收敛的较优解[13]。

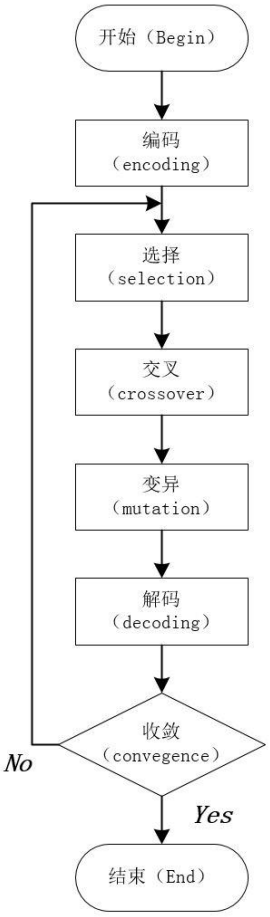

图 1 演化算法流程图

图 1 概要地说明了演化算法的基本流程，演化算法的设计核心就是解的编码形式和适值评估函数的设计。演化算法首先根据不同的问题要求将解的形式进行编码，然后种群中的每个个体代表一个问题的解。同时对于每个问题的解，还要设计评估这样的解的适值评估函数，不同的解的适值评估函数值用来衡量一个个体的适应性，适应性高的个体更容易在种群的演化中生存。演化算法的操作有初始化种群，交叉，变异，选择。初始化种群指的是随机地产生第一代种群；交叉是将亲代的个体杂交，生产新的子代种群；变异是将个体的某些特定基因进行变异，变异后的个体参与种群杂交，维持整个种群的物种多样性，同时也会得到一些新的解；选择是对种群的每一个个体进行选择，根据优胜劣汰的自然法则来淘汰劣势个体。演化算法最后通过种群的进化得到我们要获得的较优解。

3 项目管理问题

3.1 问题背景

项目管理问题中的核心点是工作包调度和资源分配。

图 2 软件开发流程

图 2 展示了一个典型软件项目在设计阶段的相关工作包管理的甘特图。通常，软件项目从最初的项目范围规划到项目中期软件的设计开发，以及到最终的项目部署和工作结束总结回顾，这些阶段都离不开对大大小小的工作包的分配。这些工作包相互之间往往有着相互制约的关系，有些工作包必须在另外一些工作包完成时才可以开工，例如分析软件需求往往要在在项目范围规划完成之后才能进行。有些工作

包可以同时进行，相互之间不会影响，例如软件开发和软件测试往往是可以同步进行的。作为一个软件开发团队的统筹规划者，面对这些大大小小的工作包，项目管理者追求的目标就是使用最短的总工期出色地完成软件项目的每一项工作包。

3.2 问题定义

本文的目标是发现下面陈述的项目管理问题的一个最优解或者近似最优的较优解：

对于一个软件项目中给定N个工作包和M种资源，这些工作包之间存在相应的依赖关系，同时每种资源只可以分配到指定的工作包使用。要求在满足工作包依赖关系和资源分配限制的前提下，合理安排工作包操作次序，使得总体工期尽量缩短。

3.3 问题描述

3.3.1 假设条件

假设一：软件项目可以被分解成一个包含 N 个元素的工作包集合 $T=\{t_1,t_2,\ldots,t_N\}$，集合 T 中每个工作包都是一个不可分割的（即在执行一个工作包过程中不能将工作包的安排计划继续拆分），集合中的工作包都有一个事先估计的工作量，这些工作量组成集合 $E=\{e_1,e_2,\ldots,e_N\}$。函数 ET: T→E，对于给定一个工作包 t_i，产生一个工作包 t_i 的估算的工作量 $e_i=ET(t_i)$，这里的 e_i 是 t_i 的估计工作量。

假设二：软件项目中的工作包可以用 M 种资源来处理，这 M 种资源组成一个资源集合 $R=\{r_1,r_2,\ldots,r_M\}$，对于每个项目的工作包集合 T 和资源集合 R，存在一个映射函数 TR：T×R→{0,1}，对于给定的工作包 t_i 和资源 r_j，$TR(t_i,r_j)=1$ 表示工作包 t_i 可以分配到资源 r_j，若 $TR(t_i,r_j)=0$ 则表示不能分配。

假设三：软件项目的工作包集合 T 之中的所有工作包，在执行过程中可能存在相互的依赖关系，这些依赖关系组成一个集合 $Dep=\{ t_i\rightarrow t_j \mid t_i,t_j\in T, t_j \text{ depends on } t_i\}$。作为本文的假设，$t_i\rightarrow t_j$ 表示 t_j 依赖于 t_i，也就是说工作包 t_j 必须要安排在工作包 t_i 的后面执行，即满足公式 $t_j.start\leqslant t_i.end$（其中 t.start 和 t.end 分别表示工作包 t 的执行的开始时间和执行的结束时间）。并假设工作包之间不存在直接或者间接的“循环”依赖关系。

3.3.2 求解目标

针对假设一，假设二和假设三，我们需要得出一个工作包安排序列：

$$S = \{ (t_{p1},r_{q1})\ldots\rightarrow(t_{pi},r_{qj})\rightarrow \ldots(t_{pN},r_{qN}) \mid t_{pi}\in T,\ r_{qj}\in R\wedge\forall i_1\neq i_2, t_{i1}\neq t_{i2} \}$$

其中，工作包安排序列 S 的每个(t_p,r_q)表示资源 r_q 被分配到 t_p。并且工作包安排序列需要满足下面两个限制：

1. $\nexists i<j, t_j\rightarrow t_i\in Dep$ ，即序列中的工作包满足依赖关系；
2. $\nexists i,k, TR(t_i,r_k)=0$，即资源可以分配给相应的工作包。

对于产生的工作包安排序列 S，每个工作包 t 就被赋予了执行的开始时间 t.start 和执行的结束时间 t.end。

我们定义项目的总工期函数为：$f(S)=\max\{t.end \mid t\in T\}$，即总工期 f 表示一个工作包安排序列 S 的所有工作包执行的结束时间的最大值，也就是说项目的最后一个工作包完成后，整个项目就结束。

项目管理问题的优化目标是在满足依赖限制和资源限制的前提下，求出一个合理的工作包安排序列 S，使得总工期最小化：

$$\textbf{Minimum}_{T,R,E,Dep,TR}(f)$$

满足：

$\nexists i<j,\ t_j\rightarrow t_i\in Dep$ （依赖限制）

$\nexists i,k,\ TR(t_i,r_k)=0$ （资源限制）

3.4 问题举例

该小节列举一个简单的项目管理问题的示例，原始项目文件见图 3，该项目被分为 4 个阶段一共有 10 个工作包，每个工作包的估计工期与所需的资源见图的相应栏目。在对于原始文件进行预处理过后，经过提取的到这个项目共有 9 个可以安排的工作包，组成集合 $T=\{t_i \mid i=1,2,3,\cdots,9\}$，并

且这个项目中有对应 4 种资源 R={r_i|i=1,2,3,4}，其中 r_1 是当一个工作包没有资源分配时，使用 r_1 作为默认分配的资源来处理这个工作包，这样能够保证所有工作包都能执行。

任务名称	工期	资源名称
Project	9 days	
Phase1	8 days	
Task1.1	1 day	Resource1,Resource2
Task1.2	7 days	Resource3
Task1.3	3 days	Resource1
Phase2	5 days	
Task2.1	1 day	Resource2
Task2.2	5 days	Resource3,Resource2
Task2.3	0 days	Resource3,Resource1
Task2.4	2 days	Resource2
Phase3	3 days	
Task3.1	3 days	Resource2
Task3.2	3 days	Resource3
Phase4	9 days	
Task4.1	9 days	Resource1,Resource3

图 3 示例项目管理工程

通常如图3所示的软件项目需要预先进行处理，经过预处理后，我们可以得到工作包的依赖关系图（见图 4）和每个工作包的估计的工作量集合 E，如工作包 t_1 的工作量为 9 天，这个预估工作量表示如果将工作包 t_1 分配到一种资源来执行这个工作包，需要 9 天才能完成这个工作包的相关操作任务。另外，图 4 还定义了工作包集合 T 的依赖关系，如 $t_5 \rightarrow t_6$ 表示工作包 t_6 依赖于 t_5。

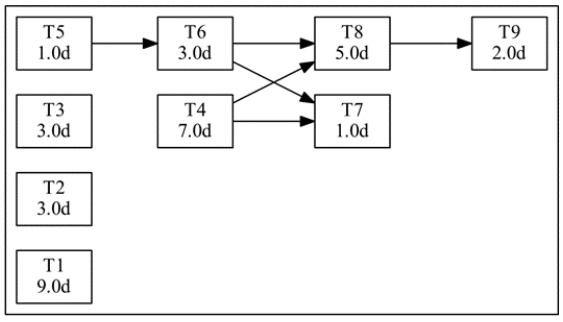

图 4 工作包依赖关系图

经过预处理后，还可以提取到的信息除了工作包依赖关系图以外，另一个重要的信息就是资源分配矩阵，即 TR 矩阵（见图 5），资源分配矩阵实际上是一个 0-1 矩阵，它描述的是 TR 函数所表示的资源分配信息。例如：TR(t_1,r_1)=0 说明资源 r_1 可以被分配到工作包 t_1 上。

	t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8	t_9
r_1	0	0	0	0	0	0	0	0	0
r_2	1	0	0	0	0	0	1	0	1
r_3	0	0	1	1	1	1	0	0	1
r_4	1	1	0	0	1	0	0	1	0

图 5 资源分配矩阵

4 算法设计

4.1 算法概要

对于上面给出的项目管理问题，本文使用演化算法来求解。使用演化算法的两个重要的步骤就是对问题解进行编码以及选取合适的适值评估函数。图 6 展示了通过模拟方法来给一个项目中所有工作包分配资源的过程，首先将项目 WBS 分解后的工作包经过预处理得到相应工作包优先安排序列，然后根据工作包依赖关系以及资源分配限制来安排每个工作包进度，最后就可以计算模拟的总工期。

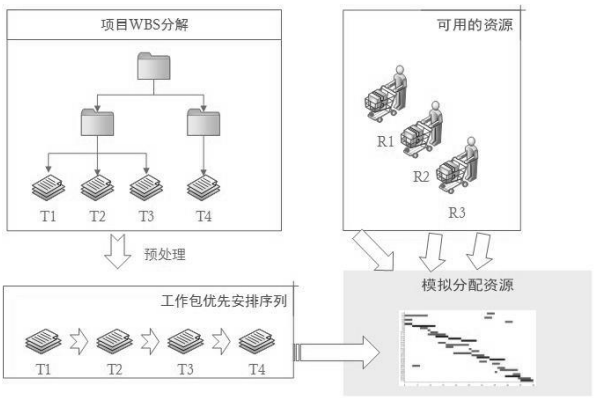

图 6 工作包分配过程

4.2 解的形式

对于上述问题的求解过程，先要确定解的形式。本文使用工作包优先序列（Work Package Priority，以下简称为 WPP）来表

示项目管理问题的一个解。这样的解的形式其实就是所有项目中的工作包的一个排列，对于包含 N 个工作包的项目的解空间就包含 N!个解，这是一个非常大的搜索空间。

WPP	1->5->6->4->8->3->9->2->7

图 7 工作包优先序列编码

图 7 就是一个特定的解，这个工作包序列编码的解给出了工作包安排的优先次序依次为 T_1、T_5、T_6、T_4、T_8、T_3、T_9、T_2、T_7。这种编码方式有利于编程，可以很好地为后面最优解的选取提供有利的手段。

4.3 适值评估

通过上面解的形式编码，每个个体的编码形式就是一个 WPP。我们通过模拟分配工作包序列的方法来模拟 WPP 分配相应的资源得到的一次工作包安排序列 S，然后计算这个工作包安排序列 S 对应的总工期，以这个总工期的值作为对种群中的一个个体的适值的评估。

对于一个 WPP 对应的总工期的求解如下，首先根据资源关系图中的各个工作包所需要分配的资源种类以及每个工作包的工作量来对每个个体工作包编码的工作包优先序列来按照 WPP 的优先级来进行依次资源分配，分配后的每个工作包就有执行的开始时间和结束时间。然后所有工作包的最大的执行结束时间就是项目的总工期。

对于一个给定的工作包优先执行序列，我们按照先来先服务的原则来给序列中的优先级每个工作包分配相应的资源，对于一个 WPP 优先级靠前的工作包可以优先占用的该种资源，获得更早的执行开始时间，优先级靠后的工作包则必须等待所有需求的资源释放后才能开始执行。通过模拟的方法求出的一次工作包安排得到的最大的资源被占用的工期就是整个项目的总工期，最终实现的工期计时器的细节具体参考算法 1。

算法 1 工期计算- 先来先服务分配算法

```
INPUT: WPP          /* work packages priority */
       R,           /* resource set */
       ET,          /* effort function */
       TR           /* TR function */
OUTPUT:  duration /* overall duration */

FOR i = 1 to WPP.length DO
  t = WPP[i];      /* pick a work packages */
  effort = ET(t);  /* get effort of work packages */
  /* initialize occupy of all resources */
  FORALL r IN R DO
    r.occupy = 0;
  END
  /* allocate resource for work package */
  FORALL r IN R DO
    /*if r can be assign to r*/
    IF TR(t,r) == TRUE THEN
      t.start = r.occupy;
      t.end = r.occupy + effort;
      r.occupy = t.end;
      BREAK;
    END
  END
END
/* get maximum of end-time for all work packages */
duration = Max(t.end);
RETURN duration;
```

4.4 遗传操作

演化算法的操作主要有两种，交叉和变异。在本文的基于序列的解编码形式下的交叉和变异方法和通常的基于二进制字符串编码有所不同。本文中的编码形式进行了重新设计。

4.4.1 交叉操作

交叉操作的目的是使用两个亲代个体生成两个子代个体。交叉的输入是两个亲代个体。本文中的交叉采取的是两点交叉，即随机选取亲代个体中的两点，互相交换染色体得到子代个体。

4.4.2 变异操作

变异操作是将个体随机交换生成一个

种群中可能没有出现过的新个体。本文中使用的变异方式是基于序列的两点交换的方式，即变异的输入是一个个体，然后在个体中随机选取两点，交换解的顺序形成变异后的新个体。

4.4 并行化演化算法

通常，并行演化算法在 GPU 平台上是可以使用 CUDA 编程接口实现[14]。本文遵循相应的设计方案，它的主要思路是利用 GPU 这样的一种众核处理器具有成千上万的计算核，在编程计算时，我们可以将种群中的每个相对独立的个体计算的任务交付到 GPU 的核上进行计算。这样并行地计算种群中每个个体时所需要的时间将会远远小于串行的计算手段。

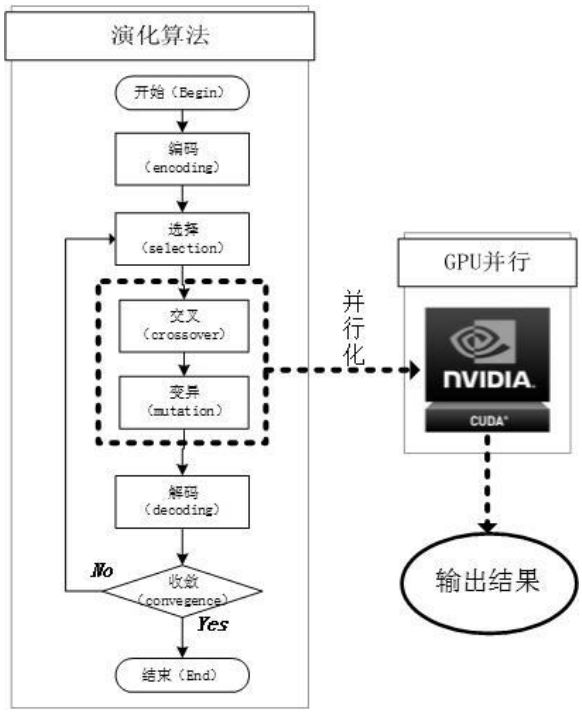

图 8 并行化演化算法

如图 8 所示，本文通过对 CUDA 的内核函数进行编程实现，可以将并行化算法中种群中的每个模拟的个体的交叉和变异操作的计算任务放到 CUDA 核上。本文在安排 GPU 实现并行算法的同时有效地利用了显卡的全局存储和共享存储的传输带宽差异，将主要的访存的操作放在了共享存储中进行。同时对于每个线程的计算资源进行了有利地调控，尽量达到负载的均衡[15]。

5 实验及结果分析

对于本文中提出的演化算法的有效性的评估。本文做了分别评估并行演化算法效率评估实验和演化算法在项目管理问题计算有效性评估实验。实验的目的是为了回答下面两个研究问题：

RQ1：并行演化算法是否能够提高工期安排的计算效率？

RQ2：演化算法是否能有效地优化项目管理问题？

5.1 实验数据

本文中评估的实验数据都是真实的工业项目管理的案例。软件项目策划一共有三个，分别命名为 A-Input，B-DBUpgrade 和 C-SmartPrice，其中 A-Input 是模拟构造的中等规模的项目策划；B-DBUpgrade 是 Oracle 数据库升级的项目策划，它的目的是一个将 Oracle 数据库从 9g 版本升级到 10g 版本，这是 Oracle 公司非公开的内部版本的项目策划；C-SmartPrice 是一个中等规模的供应链的升级的项目策划，同样的这也是一个非公开的内部项目策划[4]。数据的相关信息统计见表 1。

表 1 实验项目信息统计

	工作包数	资源种数	依赖数
A-Input	33	4	35
B-DBUpgrade	106	8	105
C-SmartPrice	74	14	73

5.2 算法效率评估实验

为了回答 RQ1，本小节是实现了并行演化算法与串行演化算法进行效率对比测试。本实验评估搭载的硬件环境是：串行演化算法的运行环境是 Intel i7 系列中央处理器（简记为 CPU）。并行演化算法的运行环境是 NVidia GeForce 系列图形处理器（简记为 GPU）。CPU 和 GPU 的详细配置对比见表 6.1。本实验搭载的软件环境是：Window 7 旗舰版 Service Pack 1，Microsoft Visual Studio 2012 C++ 编译器，CUDA 7.0.28 Runtime 编译环境。具体的硬件配置见表 2。

表 2 CPU 和 GPU 硬件配置

	CPU	GPU
型号	Core i7 CPU 870	GeForce GTX 970
主频	2.93GHz	1.27GHz
核数	8 cores	1664cores

本实验是使用上述三个项目作为输入来评估并行化演化算法和串行化演化算法的性能对比。具体代码实现上完成了两个版本的项目管理的算法的编写工作，串行版本是使用 C++语言编写的，并行版本是使用 CUDA-C++语言编写的，通过对这两个版本的代码设计实验验证并行化演化算法程序设计与串行化编程的在项目管理问题上的区别。

两个版本的演化算法的初始配置是相同的。种群的数量为 1000，演化的迭代次数为 100。每次迭代过程中通过交叉、变异和选择，在一次演化算法开始执行时设置计时器，通过计时器来对 100 次迭代过程进行计时，然后在 100 次迭代结束过后停止计时器。这样就可以统计项目问题在串行 CPU 环境和并行 GPU 环境下的分别使用的总计算时间，使用这个计算耗时作为评估标准。

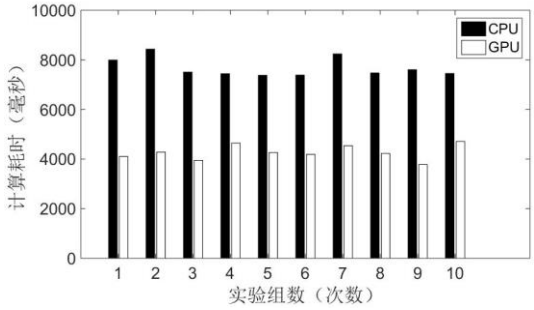

图 9 项目 A 计算耗时图

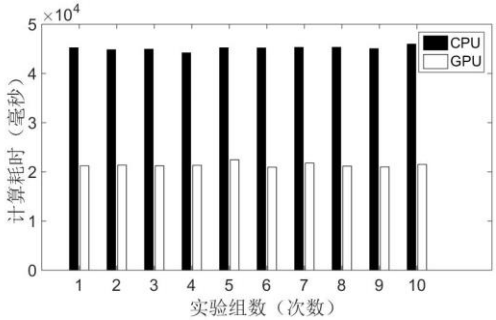

图 10 项目 B 计算耗时图

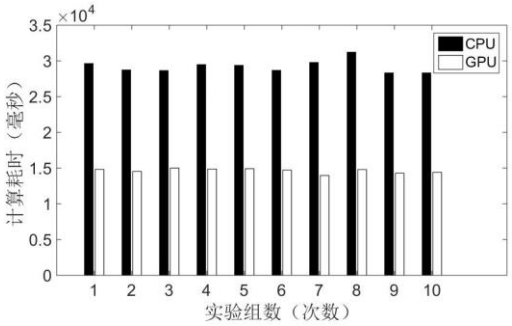

图 11 项目 C 计算耗时图

项目 A、项目 B 和项目 C 的 10 次计算耗时统计图见图 9、图 10 和图 11。可以从图中看出 CPU 上的算法耗时大体上是在 GPU 算法耗时的两倍。同时，我们在项目管理问题的计算上使用了 GPU 算法的优点是可以更高地利用系统资源。因此可以很好地回答 RQ1，并行演化算法在项目管理问题上可以有效地提高计算效率。

5.3 算法有效性评估实验

为了回答 RQ2，本小节设计了演化算法有效性评估实验实验。有效性评估实验中我们使用的基本配置是种群的个体数量为 100，演化算法的迭代次数为 100。对于每个演化算法的计算过程，我们对于种群中的每一代的所有适值进行统计，然后将离散的适值统计结果画成盒图。

盒图的每一代展示了种群中所有个体适值的五个特征值：最小值、下四分位数、中值、上四分位数和最大值。我们将输入的三个项目的每代种群适值的计算结果对应的盒图画出来，见图 12、图 13 和图 14。由三个项目变化的趋势可以看出，通常种群中初始化的时候种群的适值较为多样化，盒图的跨度较大，随着迭代的计算的次数的增加，种群向着较优的结果来演化，同时总体的适值开始降低，最后近似收敛到一个较优的结果。

通过上面实验数据分析结果，可以准确地回答 RQ2。在项目管理问题上，演化算法确实有着很好的优化提高的空间。

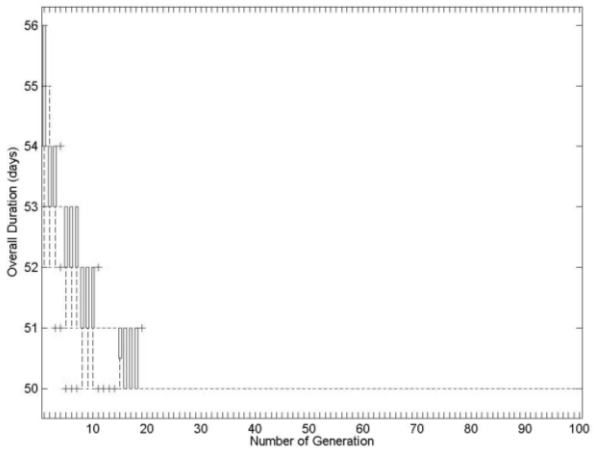

图 12 项目 A 演化趋势

图片结果显示第 19 代优化结果收敛

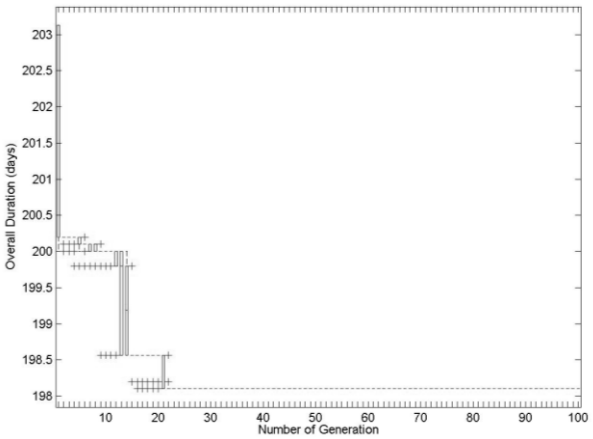

图 13 项目 B 演化趋势

图片结果显示第 22 代优化结果收敛

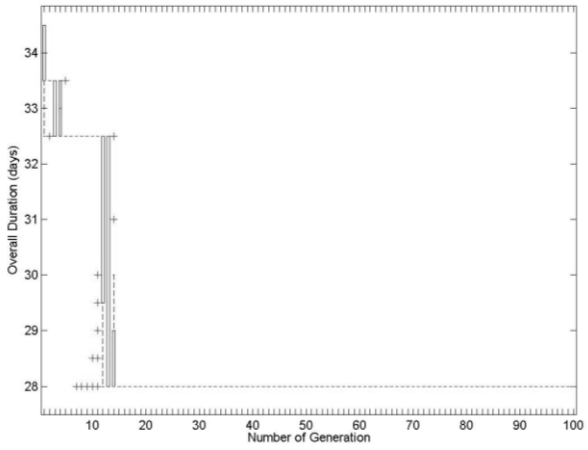

图 14 项目 C 演化趋势

图片结果显示第 15 代优化结果收敛

参考文献

[1] A. Stellman, and J. Greene. Applied Software Project Management [M]. OREILLY, 2005

[2] C. K. Chang, H. Jiang, Y. Di, D. Zhu, and Y. Ge. Time-line Based Model for Software Project Scheduling with Genetic Algorithms [J]. Information and Software Technology, 2008:1142-1154.

[3] E. Alba, and C. J. Francisco. Software Project Management with GA [J]. Information Sciences, 2007: 2380-2401.

[4] J. Ren, M. Harman, and M. D. Penta. Cooperative Co-evolutionary Optimization of Software Project Staff Assignments and Job Scheduling [C]. Lecture Notes in Computer Science, Springer Berlin Heidelberg, 2011: 495-519.

[5] D. W. Pentico. Assignment Problems: A Golden Anniversary Survey [J]. European Journal of Operational Research, 2007: 774-793.

[6] M. D. Penta, M. Harman, and G. Antoniol. The Use of Search-based Optimization Techniques to Schedule and Staff Software Projects: An Approach and an Empirical Study [C]. Software: Practice and Experience, 2011: 495-519.

[7] P. Pospichal, J. Jaros, and J. Schwarz. Parallel Genetic Algorithm on the CUDA Architecture [C], Lecture Notes Computation Science, 2010: 442-451.

[8] 曲婉玲, 刘田. 算法设计与分析[M]. 北京:清华大学出版社, 2011: 205-207.

[9] C. Stylianou, S. Gerasimou, and A. S. Andreou. A Novel Prototype Tool for Intelligent Software Project Scheduling and Staffing Enhanced with Personality Factors [C]. IEEE 24th International Conference on Tools with Artificial Intelligence. 2012: 277-284.

[10]S. Samanta. Genetic Algorithm: An Approach for Optimization (Using MATLAB) [J]. International Journal of Latest Trends in Engineering and Technology, 2014: 261-267.

[11]J. H. Holland. Adaptation in Natural and Artificial Systems: an Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence [M]. U Michigan Press, 1975.

[12]M. Harman, P. McMinn, J. T. D. Souza, and S. Yoo. Search Based Software Engineering: Techniques, Taxonomy, Tutorial [C]. Empirical software engineering and verification, 2012: 1-59.

[13]K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. A Fast and Elitist Multiobjective Genetic Algorithm: NSGA-II [J]. IEEE Transactions on Evolutionary Computation, 2002: 182-197.

[14]P. Vidal, and E. Alba. A Multi-GPU Implementation of a Cellular Genetic Algorithm [C]. 2010 IEEE World Congress on Computational Intelligence, 2010: 1-7.

[15]D.B. Kirk, W. W. Hwu. 大规模并行处理器编程实战（第二版） [M]. 北京: 清华大学出版社, 2013: 1-31.