Approximate Algorithm for Solving the General Problem of Scheduling Theory With High Accuracy

Vardan Mkrttchian, HHH University, Brisbane, Australia

Safwan Al Salaimeh, Aqaba University of Technology, Aqaba, Jordan

ABSTRACT

An approximate iterative algorithm is described, allowing the solving of general problems in the theory of schedules. The theoretical and experimental characteristic of the new algorithm are given, as well as ways of finding out its variable indicators by using parallelization computing and the implementation of a parallel version of the algorithm microprocessor. In this way, the authors have described a practical approximate algorithm for time, to solve the general problem of scheduling theory with an average error of 8% less.

KEYWORDS

Algorithm, Approximate, Implementation, Indicators, Processing Time, Properties, Representation

INTRODUCTION

To find out the problem properties as an extreme object, and a convenient representation of solving algorithms we describe in the language of graph. An approximate iterative algorithm is described, allowing solving general problems of the theory of schedules. The theoretical and experimental characteristic of the new algorithm are given, as well as ways of finding out its variable indicators by parallelization computing and implementation of parallel version of the algorithm microprocessor IBM. The described algorithm for construction an acyclic graph allows one to obtain a restriction on the number of graphs \( G \in \Omega \), not guarantee that among them will be an extreme graph \( G' \). The universal procedure that provides a basic opportunity to build all graphs \( G \in \Omega \), are given bellow

BACKGROUND

Experimental studies of algorithms KN, NZ, KP, first of all, were taken to determine the following their characteristics:

DOI: 10.4018/IJSI.2019100104
1. Average error $\Delta$ and average counting time $\tilde{T}$ as task size function $N$ at $n/m - \text{const}$;

2. Average errors $\Delta$ and average counting time $\tilde{T}$ as a function of the relationship $n/m$ at $N = \text{const}$.

In other works, the task was to confirm or reject the properties of algorithms, obtained previously in an analytical way.

The proposed algorithm is using to increase the speed of the algorithm KN, it can reach it, parallelization of computations and implementing a parallel version of computation multiprocessor. The elements of parallel computation are explicitly contained in third steps of this algorithm.

**FORMALIZATION AND PROPERTIES OF THE PROBLEM**

We formulate the general problem the traditional supply. Given finite sets of objects $I \{i \mid 1 \leq i \leq n\}$ and processing of their machines $Q \{q \mid 1 \leq q \leq m\}$ object processing is I prescription of the operation sequence $I_i = \{iq_j \mid 1 \leq j \leq j_i\}$, specific for this item. Operation $iq_j \in I_i$; executed by the machine $q \in Q$ during $t_{iq_j} > 0$, without interrupts and can begin after the completion of the immediately preceding operations $iq_{j-1} \in I_i$. Any machine $q \in Q$, only one operation can be performed at time. In view of this, all the sets of operation $I_i = \bigcup_{q \in I} I_i$ break up into $m, 1 \leq m \leq n$ classes $I_i^q, 1 \leq q \leq m$ - first operation, performed by each machine. The task of scheduling is for each machine $q \in Q$ specifies the order of the operation from the class $I_i^q$ and thus determine such months their beginning, which would provide processing a lot of all items $I$, in the minimum time. In view of the fact, that any operation $iq_j \in I_i$ uniquely identify the item number $i \in I$, sequence of operation, defined on sets $I_i^q$, $q \in Q$, represent the permutations of the number of objects on the machines. In this way, the meaning of the problem is to determine the set of permutations of machine number, minimizing the total processing time. To find out the problem properties as an extreme object, and a convenient representation of solving algorithms we describe in the language of graph. To this end, each linear order sequence of operations $I_i, i \in I$ we will represent a sequential graph $G_i = \langle I_i, V_i, W_i \rangle$ with many vertices $I_i$, arc $V_i$ and functions $W_i : I_i \rightarrow R^+$ so, every vertexes $iq_j$ this graph corresponds to one and only one operation $iq_j$ subject $I_i$, each pair of vertices $(iq_j, iq_{j+1} \in I_i, j = 1, 2, \ldots, j_i)$ arc connected $g_{ij} \in v_i$, which come from the top $iq_j$ and goes to the top $iq_{j+1}$; function $W_i$ is defined as $W_i(iq_j = t_{iq_j})$. Then a finite acyclic graph $G_i = \bigcup_{i \in I} G_i = \langle I_i, V_i, W_i \rangle$ with many vertices $I_i$, arcs $V_i$ and functions $W_i : I_i \rightarrow R^+$ it will represent the initial data of the task. Set of vertices $I_i$ top up $i_s$, $i_F$ and put $W(\{i_s\}) = W(\{i_F\}) = 0$. We connect the top $i_s$ with outgoing arcs $V_i$ from the top $I_i$, these with $G_i$ zero stage of approach. Vertices of $I_i$ zero stage of the outcome to $G_i$ we connect it to the top $i_F$ of arcs $V_F$, directed to this vertex. As a result, we will receive a technological network.

$$G_T(I_i, i_s, i_F, V_T, W_T), \quad V_T = V_i \cup V_s \cup V_F, \quad W_T : \{I_i \cup \{i_s\} \cup \{i_F\}\} \rightarrow R^+ \cup \{\emptyset\}$$

with source $i_s$ and with source $i_F$.

The length of the critical path on this network, i.e. the value of the maximum path from the vertex $i_s$ to the top $i_F$, graph $G_T$, is defined by expression:
A Case Study of Dynamic Analysis to Locate Unexpected Side Effects Inside of Frameworks
[www.igi-global.com/article/a-case-study-of-dynamic-analysis-to-locate-unexpected-side-effects-inside-of-frameworks/126614?camid=4v1a](www.igi-global.com/article/a-case-study-of-dynamic-analysis-to-locate-unexpected-side-effects-inside-of-frameworks/126614?camid=4v1a)

A Brief Overview of Software Process Models: Benefits, Limitations, and Application in Practice
*Agile Estimation Techniques and Innovative Approaches to Software Process Improvement* (pp. 258-271).
[www.igi-global.com/chapter/a-brief-overview-of-software-process-models/100282?camid=4v1a](www.igi-global.com/chapter/a-brief-overview-of-software-process-models/100282?camid=4v1a)