Applied Soft Computing 41 (2016) 235–246

Contents lists available at ScienceDirect

Applied Soft Computing

journal homepage: www.elsevier.com/locate/asoc

Monte-Carlo randomized algorithm for minimum feedback arc set

Robert Kudelić*,[1]

University of Zagreb, Faculty of Organization and Informatics, Pavlinska 2, 42000 Varaždin, Croatia

ARTICLE INFO

Article history:
Received 4 December 2014
Received in revised form
12 November 2015
Accepted 14 December 2015
Available online 13 January 2016

Keywords:
Minimum feedback arc set
Monte Carlo
Randomization
NP-hard
NP-complete
APX-hard

ABSTRACT

When we are developing information system we must, in some way, determine the development order of its subsystems. Currently, this problem is not formally solved. Therefore, to rectify this we are proposing a solution which takes the sum of weights of feedback arcs as a criteria for determining the development order, rather than some other criteria that has not come directly from information system description. For the purpose of solving this problem we have developed, analyzed, and tested, Branch and Bound algorithm and Monte-Carlo randomized algorithm which solves the problem of Information System Subsystems Development Order in polynomial time with arbitrary probability. Also, we have determined an approximation error for developed Monte-Carlo randomized algorithm. Lastly, we have proven that the problem of Information System Subsystems Development Order is NP-hard, NP-complete, and APX-hard.

1. Introduction

During information system (IS) design there are certain steps which are rather significant and without which it is not advisable to go into IS development. CASE tools readily support these steps but there is often little automation involved. "Some of the steps that could be automated are:

- IS decomposition,
- IS subsystem development order,
- ERA model automation and data base (DB) normalization,
- equalization of existing DB using syntax forms,
- automatic DB filling with integrity and referential checks on newly defined table and referential rules." [1]

In this paper we will deal with Information System Subsystems Development Order. Therefore we will continue with problem description. When we begin to design IS it is usually a good idea to make a process/data classes matrix (P/DC matrix) in order to better understand interaction of processes. It is generally known that process is a sequence of actions which is of business importance and data class is a set of data which is of business importance. P/DC matrix is represented by a set of processes and data classes in rows and columns, respectively, with interactions marked in a body of the matrix [2]. This matrix represents interactions of entire IS which is being designed and is a starting point for IS subsystems creation. An example of this matrix can be seen in Fig. 1.

Definition 1.1. Set of processes, grouped by certain criteria, represents information system subsystem.

In the IS subsystems order, from ISs_1 to ISs_4, that is seen in Fig. 1, letters *C*, *R*, and *U* that are above the diagonal (represented by IS subsystems) are feedback arcs. Letters that are bellow the diagonal are forward arcs.

It is very common to create IS subsystems according to affinity analysis. How to do affinity analysis can be seen in [3]. In short, affinity analysis groups processes together according to data class interaction between processes. Now when we have IS subsystems, we have come to the problem we want to solve.

"Problem of IS subsystems development order (ISSDO) is situated in graph theory. It is described by directed cyclic graph (from now on graph) $G=(V, E)$ where E represents data classes and V represents IS subsystems. This graph has a weight on each arc $D(E)\geq 1$. As it can bee seen from Fig. 1 it is common for subsystems to exchange relatively high, depends upon IS, amount of data classes. At least one data class has to be exchanged between IS subsystem and IS, otherwise subsystem obviously does not belong to IS.

Graph G has cycles, which means that there is a node v in graph G where $v_i \rightarrow v_{i+1} \rightarrow v_{i+2} \rightarrow \cdots v_i$. These cycles are representing a problem when we want to determine ISSDO. Namely, it means that some subsystem A has cyclical dependency from some subsystem B and it is in question which of these two subsystems should we develop first. Obviously it is a problem to develop a subsystem which has a strong cyclical dependency from other IS subsystems. That kind of subsystem then needs data classes which are currently not available and these kind of situations should be avoided.

To better illustrate this lets look at the following example with graph G and linear order (LO). Let us define graph $G=(V, E)$ with the following matrix.

$$\begin{array}{ll} 1 \rightarrow 3[2] & 1 \rightarrow 2[2] \\ 2 \rightarrow 3[8] & 2 \rightarrow 1[1] \\ 3 \rightarrow 1[1] & 3 \rightarrow 2[3] \quad 3 \rightarrow 4[2] \\ 4 \rightarrow 2[3] & 4 \rightarrow 5[1] \\ 5 \rightarrow 2[1] & \end{array} \tag{1}$$

* Tel.: +385 42390852.
E-mail address: robert.kudelic@foi.hr
[1] Artificial Intelligence Laboratory.

http://dx.doi.org/10.1016/j.asoc.2015.12.018
1568-4946/

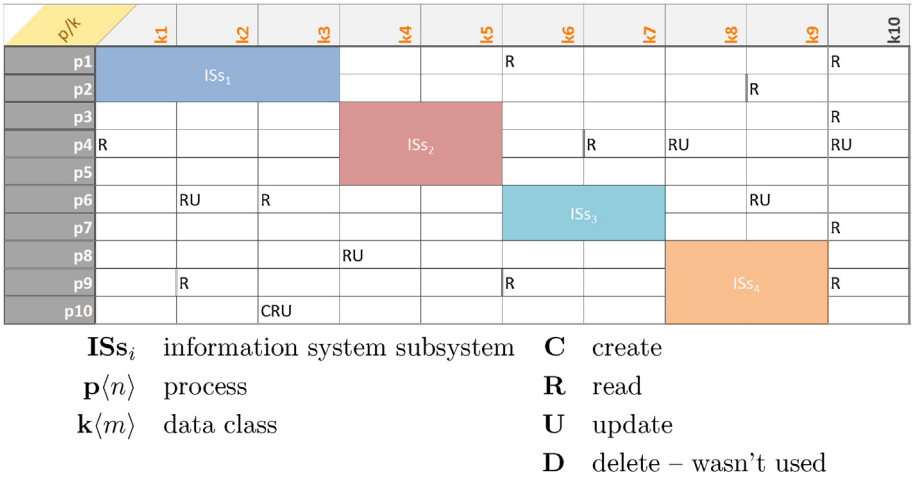

Fig. 1. P/DC matrix example with subsystems arbitrarily determined.

where $v_i \rightarrow v_j[D(E)]$ means that nodes are connected. $[D(E)]$ denotes number of arcs that are going to node v_j from node v_i (Fig. 2). Also, let us assume linear order of G; $LO(G) := (ISs_1, ISs_2, \ldots, ISs_{i-1}, ISs_i)$." [4]

$$LO(ISs_3, ISs_2, ISs_1, ISs_4, ISs_5) \tag{2}$$

Which means that we need to develop IS from subsystem 3 . . . 5. And the problem of ISSDO is now clearly seen.

Subsystem ISs_3 needs 10 data classes from the rest of the IS. That means, if we would like to develop ISs_3 first we would need 10 sets of data at our disposal or said in another way we would need, in a general case, n parts from other subsystems in order to fully develop, test, and implement ISs_3. Therefore we ask a logical question: In which order should we develop IS? It would clearly be a good thing to first develop, sequentially, subsystems that have weak dependency on the rest of IS. If subsystems are developed in parallel, what is usually the case, sequence of development represents subsystem priority in a development of the entire IS. By giving higher priority to less dependant subsystems we are guaranteeing to develop rest of the subsystems more fully. Therefore, to solve the problem of ISSDO we must find linear node arrangement with minimal sum of weights of feedback arcs.

Obviously, by developing IS this way we could potentially eliminate a lot of problems in later stages like debugging and testing, by finding bugs sooner rather than later. If we would like to find optimal solution for example for 5 subsystems we would need to calculate sum of weights of feedback arcs for 5! subsystems arrangements. Which for 5 subsystems is 120. Obviously, this is not much. But, since we are dealing with permutations, as a number of subsystems grows number of permutations grows by a factorial. This can be seen in Fig. 3. Therefore, we would like to know how hard this problem is and how can we solve it optimally and/or efficiently.

Hypothesis 1. Problem of Information System Subsystems Development Order is NP-hard and APX-hard.

Hypothesis 2. Monte-Carlo randomized algorithm solves the problem of Information System Subsystems Development Order with arbitrary probability in polynomial time.

This paper is divided into seven main sections. Section 1, Introduction, aside from the following structure of the paper and list of contributions, explains why information system subsystems development order represents a problem, gives a formal description of the problem, and states research hypotheses.

Section 2, Literature review, gives summaries of approaches that can currently be found in the literature and are related to the problem of Information System Subsystems Development Order.

Section 3, Problem of ISSDO is hard to solve, gives proofs that the problem of Information System Subsystems Development Order is NP-complete, NP-hard, and APX-hard.

Section 4, ISSDO Branch and Bound algorithm, gives Branch and Bound algorithm that optimally solves the problem of Information System Subsystems Development Order. This section consists of: formal description, pseudo-code, algorithm analysis, empirical analysis, results, and discussion of the results. Branch and Bound algorithm was developed for the purpose of determining approximation error.

Section 5, Monte Carlo randomized algorithm, gives Monte Carlo randomized algorithm that solves the problem of Information System Subsystems Development Order with arbitrary probability. This section consists of: formal description, probability calculation, pseudo-code, algorithm analysis, empirical analysis, results, and discussion of the results. *Also, this section gives main contribution of the paper.*

Section 6, Critical review, gives a comparison of our scientific results and scientific results that can be found in the literature.

And finally, Section 7, Conclusions, gives a brief overview of scientific results of the paper together with review of hypotheses.

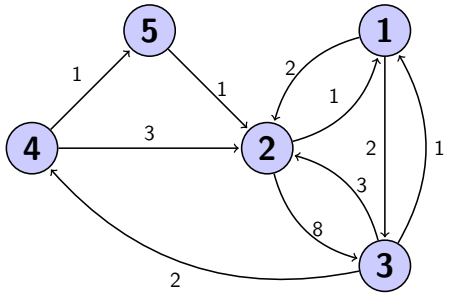

Fig. 2. Graph according to (1).

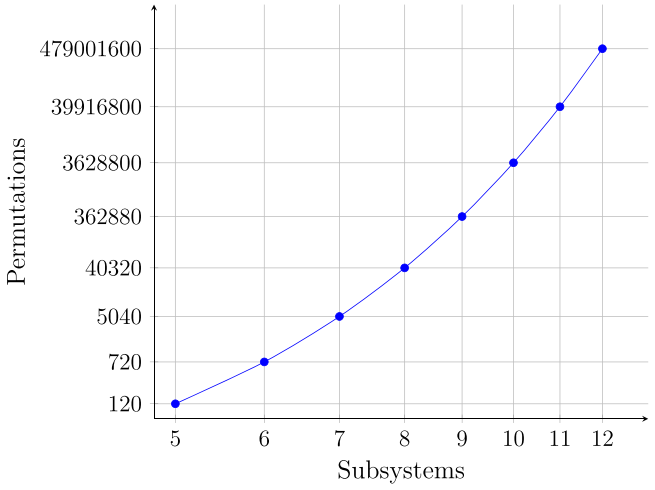

Fig. 3. Exhaustive search permutations on logarithmic scale.