

PAPER • OPEN ACCESS

## Travelling salesman problem concerning to manipulator Kawasaki FS03N trajectory formation

To cite this article: B Kulakov and D Kulakov 2019 *IOP Conf. Ser.: Mater. Sci. Eng.* **589** 012038

View the [article online](#) for updates and enhancements.

# Travelling salesman problem concerning to manipulator Kawasaki FS03N trajectory formation

**B Kulakov<sup>1,2</sup> and D Kulakov<sup>1,2,3</sup>**

<sup>1</sup>Peoples Friendship University of Russia (RUDN University), 6 Miklukho-Maklaya St, Moscow, 117198, Russian Federation;

<sup>2</sup>Bauman Moscow State Technical University, 5 Second Baumanskaya Street, Moscow, 105005, Russian Federation

<sup>3</sup>E-mail: kulakov@bmstu.ru

**Abstract:** An idea of travelling salesman problem (TSP) application to an optimal manipulator path finding was implemented. A set of known poses robot should pass is given. The poses sequence may vary. A traveling time is a performance criteria to be minimized. This task was realized concerning to a special case of Kawasaki FS03N manipulator movement by dynamic programming method. A special Bellman function state notation was used that in some cases gives an ability to solve the task at industrial robot controller. The state notation rule is one-one mapping rule between TSP state and array index it is stored.

## Introduction

In a number of cases a manipulator's goal is some object motion with specific poses it should be placed in a case when the poses order is not important – some workpiece several sides examining task for example. Herewith a necessary aim is to reduce some *additive* quality factor [1]. In that case the problem is the same with the travelling salesman problem (TSP) [2-4]. The aim is to find a time-optimal path-tracing [5-7]. The set of poses is given. Initial and final pose is set, intermediate poses should be passed single time only and interleaving order may differ. Sometimes classical TSP is formalized as follows: a complete weighted loop-free graph  $G$  is presented with vertex set  $N = \{1, 2, \dots, n\}$ ; any arc weight is nonnegative; a Hamilton circuit with minimal weight should be found [8].

This article contains a particular solution of such a task. A Kawasaki FS03N manipulator should place a workpiece mockup (just “workpiece” in further description) in fixed and known poses set (Fig. 1). Initial and final pose is the same and is labeled  $a$ .

The intermediate poses are grouped into an ordered set  $M = \{b, c, d, e\}$  (Fig. 2). The workpiece should be examined by an instrument with fixed position in space. The fastest path should be found. The intermediate poses sequence is not fixed (may vary). The manipulator should stay in each pose from among  $M$  for examination purpose during a fixed time.

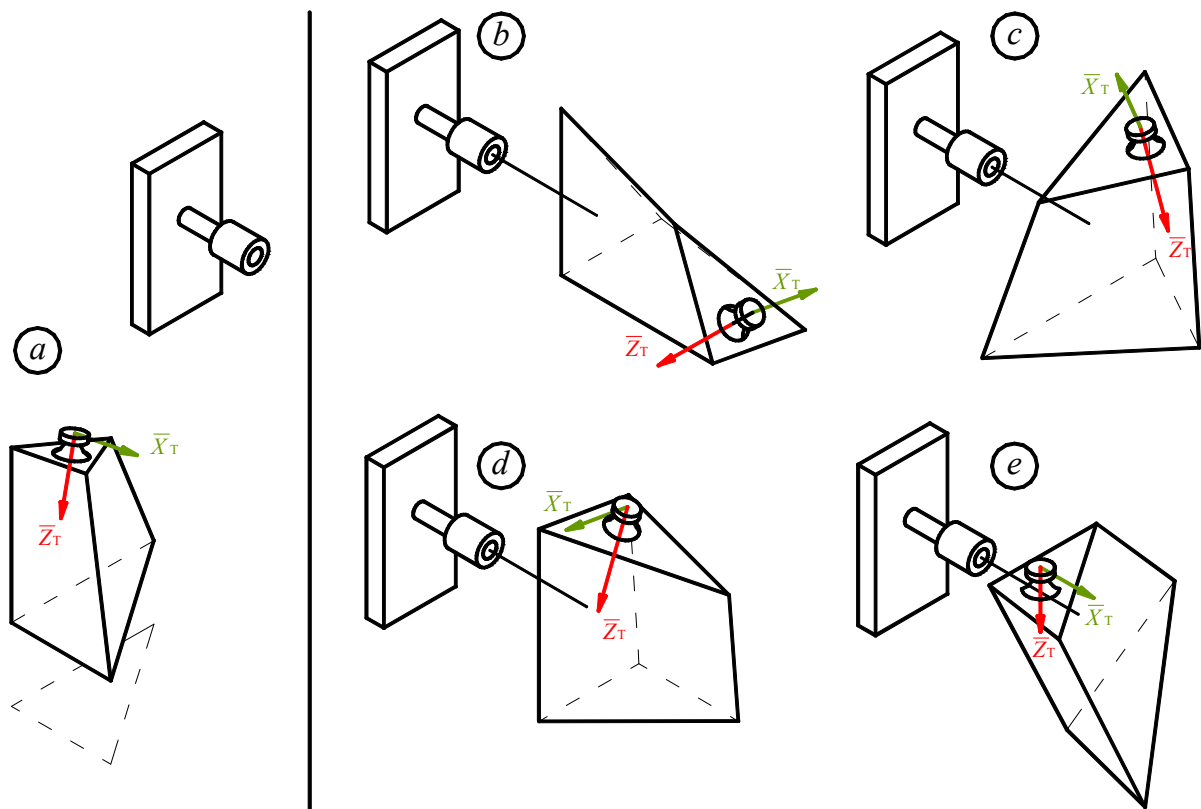
A time of detail examination – summarized time the manipulator should stay at each pose – we assume as a constant value, neglect in discussion and exclude from the travelling time.

All Kawasaki FS03N manipulator traveling time values  $T_{ij}$  (in seconds) from pose  $i$  to pose  $j$  ( $i, j = 0..4$ ) are presented in a Table 1 (for a definite motion speed). Nonzero  $i, j$  are the orders of the corresponding poses from the set  $M$ .  $i, j = 0$  correspond to the pose  $a$ .





**Fig. 1.** Kawasaki FS03N robot with the workpiece mockup in the initial pose *a*



**Fig. 2.** A schematic view of the workpiece mockup at the set of poses  $M=\{b, c, d, e\}$  and the instrument.

**Table 1.** Manipulator traveling time values  $T_{ij}$  (in seconds)  
from pose  $i$  to pose  $j$

<div style="display: inline-block; transform: rotate(-45deg); transform-origin: left top;">                     a manipulator a manipu- lator «from» pose, (i)                 </div> <div style="display: inline-block; transform: rotate(45deg); transform-origin: right top;">                     «toward» pose, (j)                 </div>	a, 0	b, 1	c, 2	d, 3	e, 4
a, 0		5.12	3.984	2.528	2.64
b, 1	5.112		6.032	5.52	5.408
c, 2	3.984	6.032		3.488	4.4
d, 3	2.528	5.52	3.488		2.704
e, 4	2.64	5.408	4.4	2.704	

All possible path variants are shown in Fig. 3. On the right of Fig. 3, opposite each terminal vertex the total traveling time is given for each case. Each vertex corresponds to some **state**: current manipulator pose and the poses it have not passed yet (a letter before the brackets and the letters in the brackets mark that correspondingly).

The task was considered and solved as the TSP with time-optimal sequence of the robot-manipulator internal poses determination. The solution we have got was realized at Kawasaki FS03N robot-manipulator. A special state notation was used to implement ease of dynamic programming realization for the assigned TSP task at the robot-manipulator.

#### State notation method for using dynamic programming for TSP

A calculated Bellman function value for almost any state should be stored at some **memory address** for further usage at dynamic programming method [9-13]. In this work the state notation is organized to compute the state memory address and do a reverse computation – define the state by the corresponding address. We use very simple rule for that:

- the state is represented by a **binary number-word** (a "0110" as an example)
- a bit order is a manipulator pose number in the ordered set  $M$ ;
- the passed poses are units, the poses are not yet passed – zeros;
- current pose is a bold unit;

For the 0110 state  $c, d$  poses are passed,  $d$  pose is current,  $b, e$  poses aren't passed yet (the initial and final pose  $a$  is not used in state recording) (Fig. 3).

This state notation in Bellman function arguments gives visually quite understandable expression as follows:

$$B(0000) = \min \{ T_{01} + B(\mathbf{1000}), T_{02} + B(0\mathbf{100}), \dots \\ T_{03} + B(00\mathbf{10}), T_{04} + B(000\mathbf{1}) \},$$

$$B(\mathbf{1000}) = \min \{ T_{12} + B(\mathbf{1100}), T_{13} + B(10\mathbf{10}), T_{14} + B(100\mathbf{1}) \},$$

$$B(0\mathbf{100}) = \min \{ T_{21} + B(\mathbf{1100}), T_{23} + B(0\mathbf{110}), T_{24} + B(010\mathbf{1}) \},$$

... etc.,

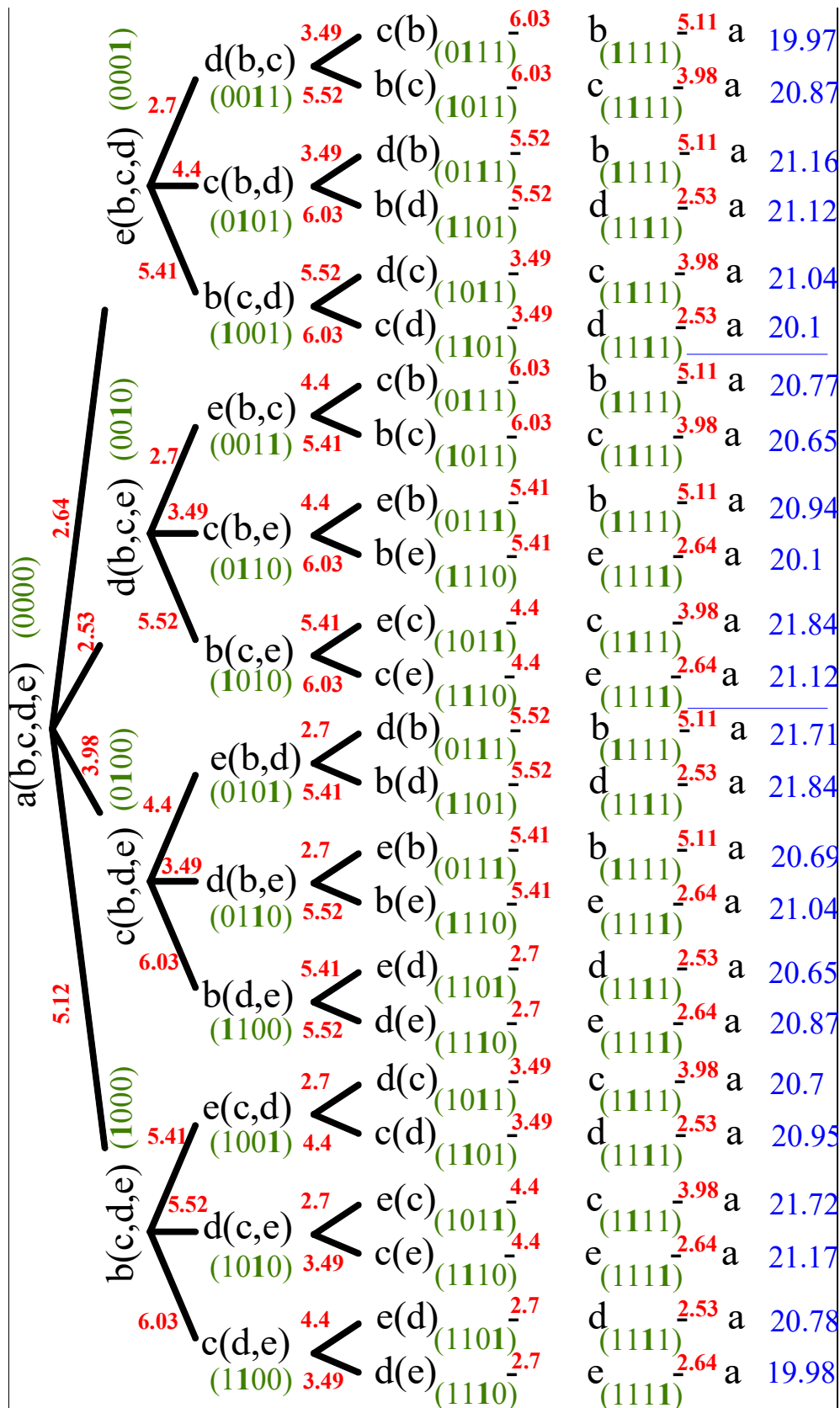


Fig. 3. Possible variants of the workpiece poses sequence during its handling time.

and also (see the Table 1):

$$B(1100) = \min\{T_{23} + T_{34} + T_{40}, T_{24} + T_{43} + T_{30}\},$$

$$B(1100) = \min\{T_{13} + T_{34} + T_{40}, T_{14} + T_{43} + T_{30}\},$$

... etc.

Here  $T_{ij}$  is a manipulator traveling time from pose  $i$  to pose  $j$  according to the Table 1.

The main **features** of the **state notation rule** for memory address computing and ability of programming usage are the follows:

– A state is described by the couple of numbers  $(p, n)$ , where:  $p$  is a passed pose index (decimal representation of the **binary number-word**);  $n$  is a bit order of the current pose (unit). For example:

$$B(0000) = B(0, 0), \quad B(1000) = B(8, 1),$$

$$B(0100) = B(4, 2), \quad B(1010) = B(10, 3).$$

etc.

– Let enter a  $Bc$  array (an array element is a traveling time from corresponding current state to final pose – i.e. a Bellman function value) and a  $Bs$  array (an array element is a next optimal pose from corresponding current state). A matrix row corresponds to passed poses, a matrix column corresponds to current pose index (a pose serial number in the set of poses  $M$ ). Then let Bellman function argument  $(p, n)$  be cell address of the arrays:  $p$  – row number,  $n$  – column number.

– With such addressing mode approximately half of all  $Bc$  and  $Bs$  array cells will be empty. For example, for the 10-th array row only the first element (state “1010”) and the third element (state “1010”) will be filled. For reduction of unused address space it is rational to use a symmetry of binary notation of serial natural numbers, demonstrated in a Table 2. Asymmetry law mathematically is shown below:

$$\text{bin}(i) = \overline{\text{bin}}(2^N - i - 1)$$

where  $i$  is a Table 2 row number in decimal representation,  $N$  is a binary digit number we use.

If only «unit cells» is filled with some information then using the described at the Table 2 symmetry we can write information to a Table 3 that don't have empty/zero cells.

The row number  $k$  of the  $Bc$  and  $Bs$  arrays will be defined as

$$k = i \quad \text{at } i < 2^{N-1}$$

$$k = 2^N - i - 1 \quad \text{at } i \geq 2^{N-1} (1)$$

**Table 2.** Binary notation symmetry of row serial number

$N_0 = i$				
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1
10	1	0	1	0
11	1	0	1	1
12	1	1	0	0
13	1	1	0	1
14	1	1	1	0
15	1	1	1	1

**Table 3.** Information record rule illustration for the case of data filling only at nonzero array cells in the Table 2

$N_0 = k$				
0	15	1	1	1
1	14	1	1	1
2	13	1	1	1
3	12	1	1	1
4	11	1	1	1
5	10	1	1	1
6	9	1	1	1
7	8	1	1	1

The  $B_c$  and  $B_s$  arrays are listed below at the Tables 4 and 5 correspondingly.

**Table 4.** *Bc* array – optimal traveling time from a current state to the final pose

№ <i>i</i>	<i>bin(i)</i>	№ <i>i</i>	<i>bin(i)</i>	<b>B</b>	<b>C</b>	<b>D</b>	<b>E</b>
<i>1</i>	<i>0001</i>	<b>14</b>	1110	–	–	–	<i>17.34</i>
<i>2</i>	<i>0010</i>	<b>13</b>	1101	–	–	<i>17.57</i>	–
<i>3</i>	<i>0011</i>	<b>12</b>	<b>1100</b>	<b>10.64</b>	<b>8.83</b>	<i>14.63</i>	<i>15.42</i>
<i>4</i>	<i>0100</i>	<b>11</b>	1011	–	<i>16.67</i>	–	–
<i>5</i>	<i>0101</i>	<b>10</b>	<b>1010</b>	<b>13.07</b>	<i>14.08</i>	<b>10.53</b>	<i>13.34</i>
<i>6</i>	<i>0110</i>	<b>9</b>	<b>1001</b>	<b>12.05</b>	<i>14.08</i>	<i>13.22</i>	<b>10.18</b>
<i>7</i>	0111	<b>8</b>	<b>1000</b>	<b>14.86</b>	–	–	–

**Table 5.** *Bs* array – optimal further pose from a current state

№ <i>i</i>	№ <i>i</i>	<b>B</b>	<b>C</b>	<b>D</b>	<b>E</b>
<i>1</i>	<b>14</b>	–	–	–	<i>3</i>
<i>2</i>	<b>13</b>	–	–	<i>2</i>	–
<i>3</i>	<b>12</b>	<b>4</b>	<b>3</b>	<i>2</i>	<i>1</i>
<i>4</i>	<b>11</b>	–	<i>1</i>	–	–
<i>5</i>	<b>10</b>	<b>2</b>	<i>1</i>	<b>2</b>	<i>3</i>
<i>6</i>	<b>9</b>	<b>2</b>	<i>1</i>	<i>4</i>	<b>3</b>
<i>7</i>	<b>8</b>	<b>2</b>	–	–	–

**The algorithm program realization**

The described above memory addressing method was tested in the algorithm for optimal sequence choice of the manipulator - workpiece poses.

During dynamic programming realization at first step an algorithm consider all the states that correspondes with any two havn't passed poses and the appropriate *Bc* and *Bs* cells are filling according the rule (1). Then at the second step all the states with three not passed poses are considered, etc.

The algorithm was realized in Matlab. Some of used built-in functions are:

- dec2bin, bin2dec, str2num, num2str, strfind – for translation between binary and decimal notations;

- nchoosek – all possible combinations of not passed poses at each step.

The algorithm work result is in the Tables 4 and 5. The cells, corresponding to  $i = 7, 11, 13, 14$  of the Tables are empty, since the corresponding states don't have variants of further trajectory.

**Analysis of results and discussion**

For better illustration of presented work the case with limit to four intermediate manipulator - workpiece poses is explored. Maximum time saving is defined by the variants with maximum and minimum traveling time. In this case it is 8.5%.

In general case many conditions affect on manipulator traveling time from one pose to another in the general case [14-17]. Some of them:

- the manipulator speed parameters;
- the instrument and base coordinate systems relative position choice [18];



- required movement accuracy;
- some type of movement parameters, etc.

In spite of above listed factors we got result that could say of:

- a manipulator path choice influence on time traveling by the given example;
- principal TSP applicability to manipulator path selection at the given conditions.

The dynamic programming state memory addressing was considered that gives some potential benefits in industrial computer realization of the TSP.

## References

- [1] Domenico Spensieri, Robert Bohlin, Johan S Carlson. Coordination of robot paths for cycle time minimization. Conference: Automation Science and Engineering (CASE), 2013 IEEE International Conference on Automation Science and Engineering 6654032, 522-527 pp.
- [2] Ereemeev, A.V., Kovalenko, Y.V. On solving travelling salesman problem with vertex requisitions (2017) Yugoslav Journal of Operations Research, 27 (4), pp. 415-426. DOI: 10.2298/YJOR161012003E
- [3] Sanjoy Dasgupta, Christos Papadimitriou, and Umesh Vazirani Algorithms, McGraw-Hill, 2006
- [4] Poznyak A.S., Najim K., Gomez-Ramirez E. Self-Learning Control of Finite Markov Chains, Marcel Dekker, Inc., New York, Basel. 2000, 298 p.
- [5] Belomestnov, V.G., Pastukhovskiy, A.V., Pervushin, N.N., Serenko, V.A. Optimization of product assembly time in a robotics complex with multiple manipulators (1993) Journal of Computer and Systems Sciences International, 31 (2), pp. 163-166.
- [6] Zacharia, P.T., Xidias, E.K., Aspragathos, N.A. Task scheduling and motion planning for an industrial manipulator. (2013) Robotics and Computer-Integrated Manufacturing, 29 (6), pp. 449-462. DOI: 10.1016/j.rcim.2013.05.002
- [7] Sethi, S.P., Sidney, J.B., Sriskandarajah, C. Scheduling in dual gripper robotic cells for productivity gains. (2001) IEEE Transactions on Robotics and Automation, 17 (3), pp. 324-341. DOI: 10.1109/70.938389
- [8] Kogan D.I. Tasks and methods of finite-dimensional optimization. Part 3. Tutorial. Dynamic programming and discrete multidimensional optimization. Publ. Nizhny Novgorod university, 2004, 157 p.
- [9] Moiseev N.N. The elements of optimal systems theory, M. Publ.: Nauka, 528 p., 1975.
- [10] Volkov I.K., Zagoruyko E.A. Operations research, series: Mathematics in technical university, M. Publ. house Bauman MSTU, 2004. 435 p.
- [11] Bellman, Richard (1954), "The theory of dynamic programming", Bulletin of the American Mathematical Society, **60** (6): 503–516, doi:10.1090/S0002-9904-1954-09848-8.
- [12] Bellman, Richard (1957), Dynamic Programming, Princeton University Press. Dover paperback edition (2003), ISBN 0-486-42809-5.
- [13] R. Bellman On the theory of dynamic programming, Proc. Nat. Acad. Sci. U.S.A. vol. 38 (1952) pp. 716-719.
- [14] Zenkevich, S., Maximov, A., Nazarova, A., Korshunov, A. Control of robot-based assembly cell (1993) Lecture Notes in Control and Information Sciences, 187, pp. 418-427. <https://www.scopus.com/inward/record.uri?eid=2-s2.0-84946075718&partnerID=40&md5=5298f3e34b5f4f2dfe79e320cd2b551c>
- [15] Nosova, N.Yu., Glazunov, V.A., Misyurin, C.Yu., Filippov, D.N. Synthesis and the kinematic analysis of mechanisms of parallel structure with the outcome of progress (2015) Izvestiya Vysshikh Uchebnykh Zavedenii, Seriya Tekhnologiya Tekstil'noi Promyshlennosti, 2015-January (2), pp. 109-113.
- [16] Leskov, A., Golovin, V., Arkhipov, M., Kocherevskaya, L. Training of robot to assigned geometric and force trajectories (2016) Mechanisms and Machine Science, 39, pp. 75-84. DOI: 10.1007/978-3-319-30674-2\_6

- [17] Leskov, A., Illarionov, V., Zimin, A., Moroshkin, S., Kalevatykh, I. Distance robotics learning using Hybrid Simulating Testbed (2014) Proceedings of 2014 11th International Conference on Remote Engineering and Virtual Instrumentation, REV 2014, статья № 6784261, pp. 225-226. DOI: 10.1109/REV.2014.6784261.
- [18] Domenico Spensieri, Johan S Carlson, Robert Bohlin, Jonas Kressin. Optimal Robot Placement for Tasks Execution. DOI: 10.1016/j.procir.2016.02.105