Cyclic scheduling of a robotic cell

Wojciech Bożejko
Andrzej Gnatowski
and Ryszard Klempous
Wrocław University of Science
and Technology,
Department of Automation,
Control and Mechatronics,
Janiszewskiego 11-17,50-372 Wrocław, Poland
{wojciech.bozejko,andrzej.gnatowski,
ryszard.klempous}@pwr.edu.pl

Michael Affenzeller
and Andreas Beham
University of Applied Sciences Upper Austria,
Heuristic and Evolutionary Algorithms Laboratory,
Softwarepark 11, 4232 Hagenberg, Austria
{michael.affenzeller,andreas.beham}@fh-hagenberg.at
Johannes Kepler University Linz,
Institute for Formal Models and Verification,
Altenberger Straße 69, Linz, Austria

***Abstract*—The paper deals with human-computer interaction in which the cooperation leads to solve a difficult issue of discrete optimization. Considered jobs scheduling problem in a robotic cell consisting of two machines and a robotic operator. Only one of two machines can work at a time and there are setup times between successive operations on a machine. The goal is to determine a schedule – permutation of jobs – and an assignment of jobs to machines, which minimize the minimal cycle time. We show that although there is exponential number of assignments of jobs to machines, it is possible to determine optimal assignment in the polynomial time. Next, we propose higher level metaheuristics – tabu search and evolutionary algorithm.**

I. Introduction

The paper deals with an issue of Cognitive Infocommunication in which a human and ICT (computer or robot) represent a kind of combo. This scenario is emerging in the paper where human and robot actually formulate a common cognitive capability in cooperation (see [2], [3]) for solve cyclic scheduling problem – a difficult issue of artificial intelligence from the field of operations research and discrete optimization. Cyclic scheduling increases its popularity in practical applications as well as in theoretical researches. Machine scheduling problems with cycle time minimization was considered by Kampmeyer [7]. Bożejko et al. propose tabu search algorithm for the cyclic machine scheduling problem [5]. Also simulated annealing metaheuristic was considered [6]. A survey of complexity of cyclic scheduling problems can be found in the work of Levner et al. [8]. In this work we consider the manufacturing/service system having two non-uniform stages (machines) working in parallel and the single operator (human, robot) dedicated to service both stages, see Fig. 1. Given the set of tasks should be processed in the system. Each task can be processed on any one of these machines with various processing time depending on the chosen machine, sequence dependent and machine dependent setup times between tasks, and requires continuous presence of operator during its processing as well as the setup activity. We would like to find the workload of machines, policy of operator, and processing order, which lead us to cyclic schedule with minimal cycle length. Furthermore, we propose a new method for generic modeling of interrelated processes which we call optimization networks. The optimization of interrelated knapsack and traveling salesman problems using this method has been shown [4]. In this particular case a machine assignment problem and a scheduling problem are proposed to be coupled to an optimization network.

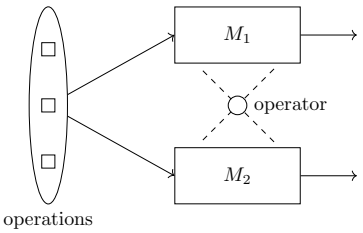

Fig. 1. Manufacturing/service system

978-1-5090-2645-6/16/$31.00 ©2016 IEEE

II. Mathematical model

List of notions:

- n – number of tasks,
- N – set of tasks; $N = \{1, 2, \ldots, n\}$,
- a_i, b_i – processing time of task i on machines M_1 and M_2, respectively,
- s_{ij}, t_{ij} – setup time between task i and task j on machines M_1 and M_2, respectively,
- x_i, – binary variable, $x_i = 0$ if task is processed on M_1 and $x_i = 1$ if on M_2,
- $x = (x_1, x_2, \ldots, x_n)$ – assignment of all tasks,
- S_i – starting time of task i,
- C_i – completion time of task i, $C_i = S_i + a_i$ if $x_i = 0$; $C_i = S_i + b_i$ if $x_i = 1$;
- π – task processing order (permutation on N),
- $T(\pi)$ – minimal cycle time for the given π,

System executes tasks in the cyclic way, namely $\pi \circ \pi \circ \ldots$, by repeating the task processing order $\pi = (\pi_1, \pi_2, \ldots, \pi_n)$ being the permutation on the set N. Each task can be directed for processed either to M_1 or to M_2. Vector x describes the chosen allocation. Given x, we can partition π into two subsequences, performed by M_1 and M_2, respectively,

$$\pi_A = \pi|_{x_i=0},\ \pi_B = \pi|_{x_i=1}, \tag{1}$$

with cardinalities $n_A = |\pi_A|$, $n_B = |\pi_B|$, $n_A + n_B = n$. Let $T(\pi)$ denote the minimal cycle time found for the given π, and $T_x(\pi)$ cycle time for the given π and x. We have

$$T_x(\pi) = \sum_{i=1}^{n_A}(a_{\pi_A(i)} + s_{\pi_A(i),\pi_A(i+1)}) + \\ + \sum_{i=1}^{n_B}(b_{\pi_B(i)} + t_{\pi_B(i),\pi_B(i+1)}), \tag{2}$$

where $\pi_A(n_A+1) = \pi_A(1)$, $\pi_B(n_B+1) = \pi_B(1)$. The optimization problem is: find π^* such that

$$T(\pi^*) = \min_{\pi} T(\pi), \tag{3}$$

where

$$T(\pi) = \min_{x} T_x(\pi). \tag{4}$$

Because TSP is a special case of (3) – (4), the stated scheduling/workloading problem is NP-hard. The proposed mathematical model is adjusted actually to a decomposition, used in the solution approach. Equation (3) defines *upper level*, whereas (4) – *lower level* problem.

III. Lower level

This section aim is to find $T(\pi)$ for the given π, as it has been stated in (4). The analysis will be carried out here assuming single-processor (sequencing) model of performing calculations. Starting from exponential ad hoc method, we will propose the original polynomial-time algorithm and then will show the reduction of the computational complexity. Further improvement of efficiency assuming parallel calculation model will be discussed in successive sections.

Without the loss of generality we assume hereinafter that $\pi = (1, 2, \ldots, n)$, since always tasks can be re-numbered in the appropriate manner. Generally, one can assume that processed tasks are indexed by consecutive integers $i = 1, 2, 3, \ldots n \ldots$ accordingly to their location on the time axis. Such indexation does not fit to the limited set N, original data, and does not reflect the cyclic character of processing. To establish the relation between these two indexing system we introduce the *reduced index* notion

$$[i] = 1 + ((i-1) \bmod n), \tag{5}$$

which transforms the former sequence into the latter. For example, for $n = 5$ the infinitesimal sequence $i = 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, \ldots$ will be converted into repetitive identity task permutations $[i] = 1, 2, 3, 4, 5, 1, 2, 3, 4, 5, 1, 2, \ldots$. We use the symbol $i : j$ to denote the subsequence of indices $(i, i+1, \ldots, j-1, j)$, $i < j$. Clearly, if these indices go beyond n, each symbol in this subsequence should be replaced by suitable reduced index before the application of data. In this point of considerations, the reader may ask about the reasons of using two index systems. The answer is 'for convenience of so called block representations' explained next. Indeed, permutation on N as well as the assignment x one can express in the cyclic manner using sequences of the length n, see Fig. 2. Unfortunately, decomposition of x into blocks hardly to explain with the use of cyclic representation, because of unknown a priori start position of blocks, which implied a complicated notation style. Therefore, we cut of cyclic representation and unroll it on the plane. To ensure considerations of float starting points for blocks for single cycle of task performing we need

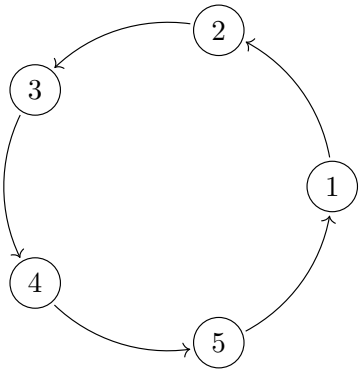

Fig. 2. Cyclic representation for $n = 5$

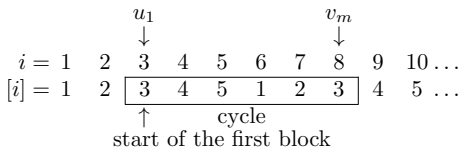

Fig. 3. Unrolled cyclic represenation

to consider two cycles, which correspond to indices $1 \leq i < j \leq 2n$, see Fig. 4.

In order to simplify denotation, we write assignment x as a binary sequence without commas, i.e. $x = (10110001 \ldots 100)$. Since two cycle repetitions will be analysed, x is repeated twice and called $X = (10110001 \ldots 100|10110001 \ldots 100)$. In X appear homogenous subsequences $(000 \ldots 0)$ and $(111 \ldots 1)$. Based on them, we distinguish two type of *blocks*, see also Fig. 3:

- A-type – maximal subsequence of X from position i to j that begins in X from single 0 on position i, next having 1 from position $i + 1$ to j (the sequence of 1 is as long as possible), which means that on position $j + 1$ is also 0,
- B-type – maximal subsequence of X from position i to $j$that begins in X from single 1 on position i, next having 0 from position $i + 1$ to j (this sequence of 0 is as long as possible), which means that on position $j + 1$ is 1.

Since x represents an assignment for N being the binary number with n bits, theoretically we have $O(2^n)$ various sequences x. The full overview of x for the given π lead us to exponential-time algorithm with unacceptably high computational complexity $O(n2^n)$. In the sequel we will show the polynomial-time algorithm $O(n^3)$ based on specific graph properties.

Taking into account the possible variety of x, we can easily analyse two extreme cases. For $x = (0000 \ldots 000)$, the minimal cycle time is equal

$$D_0 = \sum_{k=1}^{n} (a_k + s_{k,k+1}), \tag{6}$$

where $s_{n,n+1} = s_{n,1}$. For $x = (1111 \ldots 111)$, the minimal cycle time is equal

$$D_1 = \sum_{k=1}^{n} (b_k + t_{k,k+1}), \tag{7}$$

$t_{n,n+1} = t_{n,1}$. Let us consider remaining cases where x is a mixture of 0 and 1 elements. At first, we decompose x (more precisely X) into blocks of A or B type, see also Fig. 4. The sequence of successive blocks is described as $(u_k : v_k)$, $k = 1, 2, \ldots, m$, and owns the following features: (a) $1 \leq u_1 \leq n$, (b) $u_k < v_k$, (c) $v_k = u_{k+1}$, $k = 1, 2, \ldots m - 1$,(d) $[v_m] = u_1$, (e) each block is of type A or B, (e) blocks appear in the alternant order, e.g. either $ABABABA \ldots$ or $BABAB \ldots$. We define for blocks a measure called length. The *length of the A-type block* $(i : j)$, $1 \leq i < j \leq 2n$, is defined as follows:

$$A_{ij} = a_{[i]} + s_{[i],[j+1]} + \sum_{k=i+1}^{j-1} (b_{[k]} + t_{[k],[k+1]}). \tag{8}$$

The *length of the B-type block* $(i : j)$, $1 \leq i < j \leq 2n$, is defined as follows:

$$B_{ij} = b_{[i]} + t_{[i],[j+1]} + \sum_{k=i+1}^{j-1} (a_{[k]} + s_{[k],[k+1]}). \tag{9}$$

From (8) – (9) we conclude immediately that $O(n^2)$ values A_{ij} (also B_{ij}) for all $(i : j)$ inside the mentioned scope of i, j can be found in $O(n^3)$ time. Note that task $[j]$ does not contribute to A_{ij} as well as to B_{ij}.

In order to find $T(\pi)$ using blocks we create the two-layer, directed graph, reflecting all possible block decompositions, see Fig. 5,

$$G = (V_A \cup V_B, E_A \cup E_B), \tag{10}$$

where

$$V_A = \{1, 2, \ldots, 2n\}, \;\; V_B = \{1', 2', \ldots, (2n)'\}, \tag{11}$$

are sets of unweighted nodes representing task performed on M_1 (these from V_A) and performed on M_2 (these from V_B), whereas

$$E_A = \{(i, j') : \; 1 \leq i < j \leq 2n\},$$

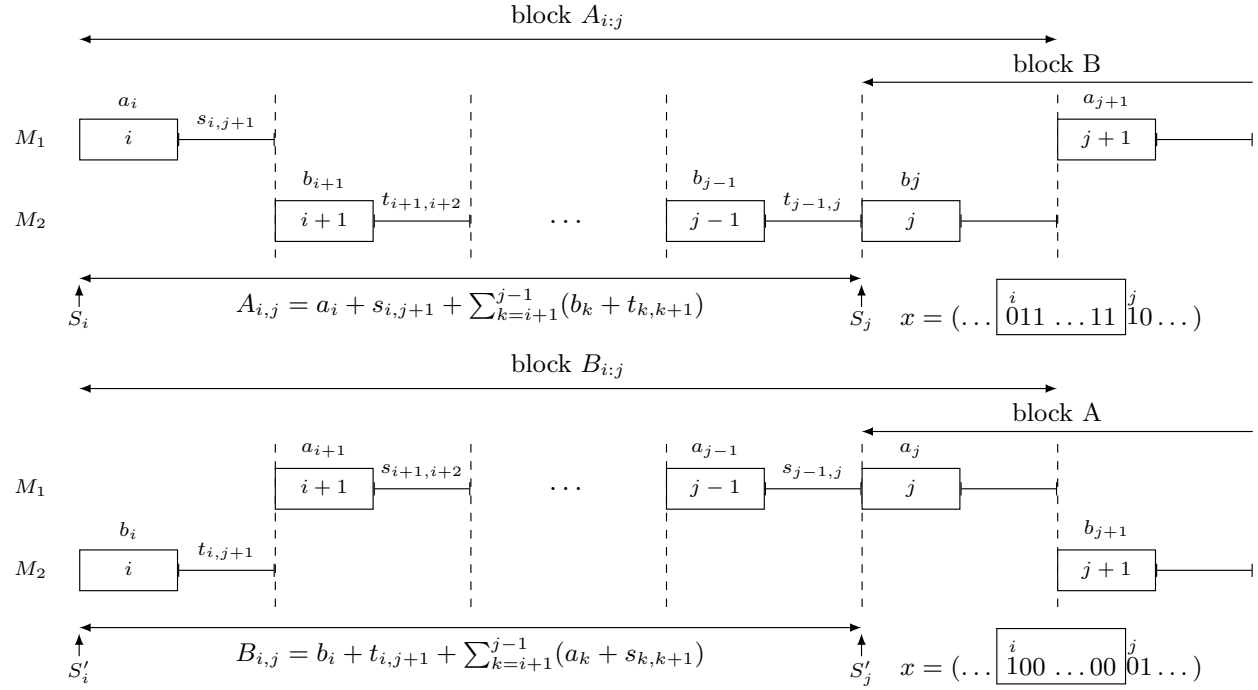

Fig. 4. Blocks and their lengths

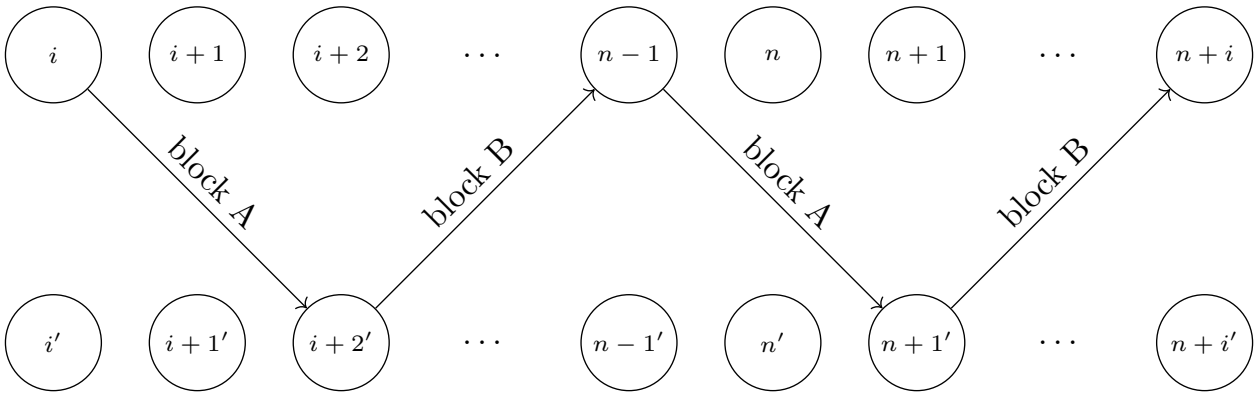

Fig. 5. Graph G

$$E_B = \{(i', j) : \ 1 \leq i < j \leq 2n\}, \tag{12}$$

are sets of arcs. Each arc $(i, j') \in E_A$ has weight A_{ij}, each arc $(i', j) \in E_A$ has weight B_{ij}. Arcs represent minimal time lag between starting time moments of tasks i and j performed in blocks $A_{i:j}$ (in case of E_A) and $B_{i:j}$ (in case of E_B).

A path in G from node i to $n + i$ represents a decomposition of T into blocks $ABAB\ldots A$, while the path from i' to $(n + i)'$ represents a decomposition of T into blocks $BABA\ldots B$. In both cases T can be interpreted as the path length, being the sum of blocks lengths. Let us denote by D_i^A the shortest path from node i to $n+i$, and by D_i^B the shortest path from node i' to $(n + i)'$. We have

$$T(\pi) = \min\{D_0, D_1, \min_{1\leq i\leq n} D_i^A, \min_{1\leq i\leq n} D_i^B\} \tag{13}$$

IV. Upper level optimization

The two problems together can be modeled as an *optimization network* whereby one solver is responsible for optimizing the job orders, as represented by a permutation, while the other solver optimizes the assignment of jobs to the machines, as represented by a binary vector. For this purpose various combinations

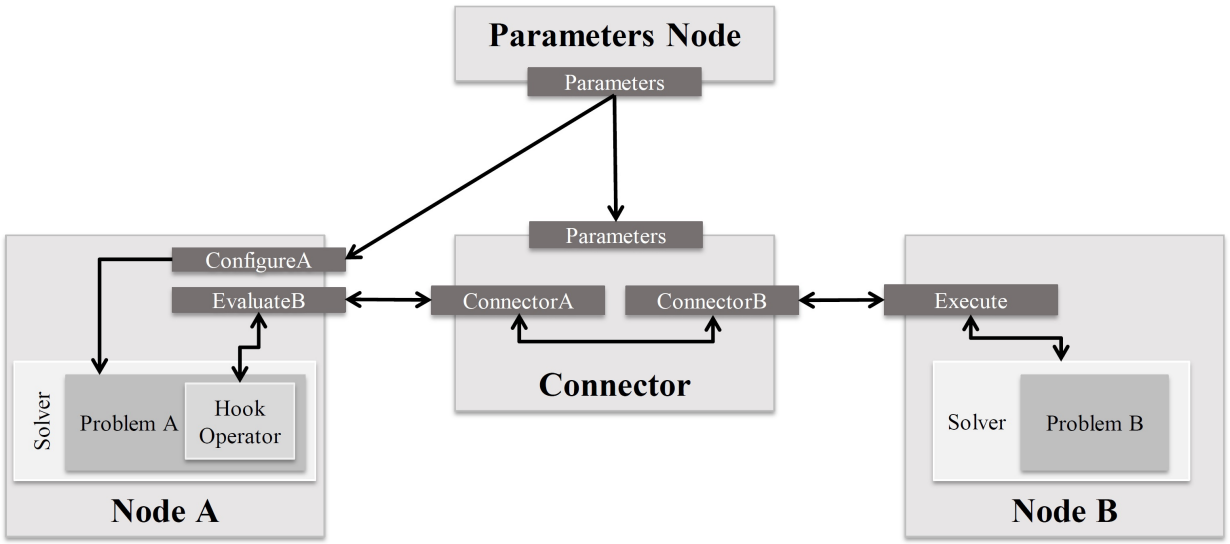

Fig. 6. Schematic diagram of the interconnection of an upper and a lower level optimization problem in an optimization network

of metaheuristic solvers can be examined on the two-layers network problem formulation (see [1]). In the network there would be two ways to model and solve the problem:

1) Using the job order as initiating problem.
2) Using the machine assignment as initiating problem.

Concerning the job order as the principal problem the task for the subsequent solver is to find an optimal – or very good – machine assignment for each given permutation. This could be performed with an efficient metaheuristic applied to optimize the permutation while the machine assignment is solved in $O(n^3)$ as described in this paper. Metaheuristic such as simulated annealing, tabu search, variable neighborhood search, scatter search, or variants of population-based evolutionary algorithms have shown to be suitable in permutation-based problems.

In the second case the machine assignment would be seen as the principal problem. In this case the jobs would be split into M groups and thus M single-machine cyclic scheduling problems with sequent-dependent setup times have to be solved - a problem that is identical to an asymmetric traveling salesman problem. This may be regarded as inefficient at first, though it has to be stated that the solution space of the binary vector is of size M^n while that of the permutation is $n!$ which would benefit from a reduction due to splitting the global job sequence into a sequence per machine.

In comparison of the two approaches the solution space of the second case is smaller despite the efficient approach of optimizing the machine assignment as described in this paper. However, while search space is an important factor of problem difficulty there are other factors such as the ruggedness of the fitness landscape, e.g. the number of local optima, the modality, isotropy, or deceptiveness. These properties are however not known a priori and would have to be studied using fitness landscape analysis. Both approaches would have to be compared against each other in empirical experiments using state of the art solvers.

V. Conclusion

We consider an NP-hard problem of determination of the schedule for a two-machine robotic cell. We propose a method of operations-to-machine assignment determination for a fixed permutation of jobs which makes it possible to find optimal assignment from the set of possible assignments, which has exponential cardinality, in the polynomial time. We also discuss optimization networks usage – the new method of interrelated processes modeling.

References

[1] M. Affenzeller, S. Winkler, S. Wagner, A. Beham, G. Kronberger, M. Kofler, "Metaheuristic Optimization", *Hagenberg Research*, pp. 103-156, Springer, 2009.

[2] P. Baranyi P., A. Csapó, "Definition and synergies of cognitive infocommunications", *Acta Polytechnica Hungarica* vol. 9 no. 1, pp. 67–83, 2012.

[3] P. Baranyi, A. Csapó, Gy. Sallai, "Cognitive Infocommunications (CogInfoCom)", Springer International, 2015.

[4] A. Beham, J. Fechter, M. Kommenda, S. Wagner, S. M. Winkler, M. Affenzeller, "Optimization Strategies for Integrated Knapsack and Traveling Salesman Problems", *Lecture Notes in Computer Science* no. 9520, Las Palmas, Gran Canaria, Spanien, pp. 359–366, 2015.

[5] W. Bożejko, M. Uchroński, M. Wodecki, "Block approach to the cyclic flow shop scheduling, *Computers & Industrial Engineering* no.81, pp. 158–166, 2015.

[6] Bożejko W., Pempera J., Wodecki M., "Parallel Simulated Annealing Algorithm for Cyclic Flexible Job Shop Scheduling Problem", *Lecture Notes in Artificial Intelligence* no. 9120, pp. 603–612, Springer, 2015.

[7] Kampmeyer T., "Cyclic scheduling problems", *Ph.D. Dissertation*, Universitat Osnabruck, 2006.

[8] Levner E., Kats V., Lopez A. P., Cheng T. C. E., "Complexity of cyclic scheduling problems: A state-of-the-art survey", *Computers & Industrial Engineering* no. 59, pp. 352-361, 2010.