A hybrid metaheuristics for layout optimization in production cell

Hugo Zupan, Janez Zerovnik, Niko Herakovic

Abstract — A local search heuristics based on “Remove and Reinsert” (RaR) neighbourhood for the arrangement of machines in production cell is proposed. The result of the optimization lowers the transport cost of the process enormously, which is obvious from the research results of this paper.

Keywords — Layout optimization, Remove and Reinsert algorithm, iterative improvement, discrete event simulation, optimization, production cell

I. Introduction

In the discussions about the “Factories of the Future” [1] one of the main objectives is to find the way to effectively manage unnecessary wastes in production. The contemporary manufacturing practice is therefore very often based on different production theories like Lean production, Just in Time, Kanban and Pull systems, based on the Toyota production system (TPS) etc., as well as on supply chain and overall equipment efficiency (OEE) optimization [2], [3], [4], [5], [6], [7]. The Lean system has become even a reference model for optimizing production and general performance of larger companies as well as SMEs [8], [9], [10]. Therefore some research works focus in simulation of production processes as well as in material flow and batch quantity optimization [11], [12], [13]. For this reason, the traditional layout of workshop machinery is transformed into more production cells, which is also one of the goals of Lean production [14]. For the formation and optimization of the cells, there has been proposed a variety of different methods [15], [16], [17], [18], [20], [21], [22]. In addition to those methods in recent years the usage of method simulation with discrete events is increasing [23], [24], [25].

It is well known that using discrete event simulation or virtual factory is very effective tool for “what-if” scenarios, for every type of production system [26], [27]. In our case we have transformed a possible real production system with all the features and limitations into the simulation model. The idea is that the metaheuristics proposes an initial and iteratively improved arrangement of machines in production a cell while the discrete event simulation performs “what-if” scenario for each proposed arrangement thus providing the quality measure of the arrangement. This process is repeated until the metaheuristics can no longer provide better arrangement of machines.

The rest of this short contribution is organized as follows. In the section II., the metaheuristics RaR is outlined and its operation is illustrated with an example in Section III. Results of the first experiments are given in Section IV and concluding remarks in Section V.

II. The metaheuristics RaR

The proposed metaheuristics is called RaR algorithm. The idea (algorithms were given various names) was successfully applied to the probabilistic traveling salesman problem (PTSP) [28], the asymmetric traveling salesman problem (ATSP) [29] and to the classical resource-constrained project scheduling problem (RCPSP) [30]. The basic idea of the heuristics is very simple, and this may be a reason for good results. It may be rather surprising that a simple heuristics outperforms much more complicated metaheuristics such as for example the genetic algorithms, but we believe that this phenomena is not that much unexpected - see [31] and the references there.

RaR can be regarded as a hybrid metaheuristics that consists of two phases: generating an initial solution, and iterative improvement. In the first phase, a solution is usually constructed in a way to solve a sub-problem optimally first, and then inserting the remaining machines, not unlike the well-known heuristics arbitrary insertion for TSP does [32]. The iterative improvement phase, roughly speaking, removes some of the machines from the current cell, and reinserts them back into the cell in arbitrary order. The arrangement of machines is known to be equivalent to the traveling salesman problem (TSP). Adaptations and modifications used for successful application to the problem studied here are explained in some detail below.

Here, we assume that the initial solution is given (or it is just a random arrangement of machines) and focuses on the iterative improvement phase.

In the inner repeat loop, a subset of $m$ machines is selected, and an optimal permutation of these $m$ machines is found by exhaustive search. It should be noted that, contrary to RaR applied to other problems, the machines which are not selected are not removed, only their relative position is frozen. After the optimal permutation of the selected $m$ machines is found, these machines are, one by one, removed and reinserted into the solution into the best position, keeping all other relative positions of machines fixed. The loop is repeated until there is no improvement, and then the position of selected $m$ machines is changed. First, the $m$ machines at positions $1...m$ are chosen, then the positions $2...m+1$, and finally the positions $n-m+1...n$ are regarded.
The RaR Algorithm

\[ S \leftarrow \text{initial Schedule (M1, M2, ..., Mn)} \]

repeat

\[ S'' \leftarrow S \]

\[ w = 1 \quad (\text{position from which the permutation starts}) \]

\[ x = 0 \]

repeat

\[ S' \leftarrow S \]

1. Choose \((w \ldots w+m-1)\) machines from \(S\) and start with permutation on initial state \(w\)

2. Check every solution for every combination from permutation table

3. Choose the best solution and place it in the \(S\)

4. If permutation found better solution than \(S'\) then \(x = 0\)

begin (if \(x = 0\))

\[ P \leftarrow \text{Optimum of first step (S)} \]

\[ Y = m + w \quad (m - \text{number of machines processed with permutation}) \]

while \(Y < n\) (number of all machines) do

\[ P' = P \]

\[ q = 1 \]

repeat

\[ P'' \leftarrow \text{Insert the machine} \]

\[ \text{form place } Y \text{ on place } q \]

\[ q = q + 1 \]

until \(q = Y\)

Find the best solution (\(P''\))

\[ P' = \text{Choose (P'')} \]

\[ Y = Y + 1 \]

endwhile

end

\[ S \leftarrow \text{Choose (P)} \]

until \(S' /= S\)

\[ w = w + 1 \]

\[ x = 1 \]

until \(w > n - m\)

until \(S'' /= S\)

\[ \]

III. Problem instance

The heuristics has been tested on a realistic example. To the best of our knowledge, there are no benchmark instances in the literature; we plan to generate a dataset of several instances for a more extensive experiment with results to be reported in the full paper.

We have a production process with 10 work places where operations are carried out. The plan is to produce 20 orders. For each order, the technology procedure and the sequence of operations are known. Raw material goes from material storage and the finished products are stored in the products storage. Orders require different operations. The transport cost between orders is assumed to be 1 per meter.

Machines are distributed in a circle with the same distance between neighbourhood machines (see Figure 1). Based on the input data (technology, quantity, material flow, costs of transportation) we designed a simulation model.

The simulation illustrates the current state of production cells and covers all its properties in a selected point of time.

Output data simulation includes:

- the intensity of the material flow between the machines,
- costs of transport and
- Sankey chart.

![Figure 1. Logical scheme of production cell model.](image)

We briefly outline the operation of the algorithm on the instance. The first step is using the permutation table. The idea is that from the whole instance, which has \(n\) machines, we take \(m\) machines (\(m < n\); we use 5) and optimize them using all possible combinations, which are enshrined in the permutation table. We do this by finding the best arrangement of where the chosen \(m\) machines are permuted and the other \(n-m\) machines’ relative positions do not change. We continue by considering the machines that were fixed in the first step. One by one, these machines are removed from the arrangement and reinserted, this time without changing relative positions of all other machines.

In our case we have initial schedule of 10 machines \((M1, M2, M3, M4, M5, M6, M7, M8, M9, M10)\). Then we choose the first \(m\) (5 in our case) machines and find their permutation with minimal cost. If there is more than one best permutation, the first is taken. In our case, the optimum after first step is: \((M2, M1, M3, M5, M4, M6, M7, M8, M9, M10)\).

In the second step we start with inserting the other machines into the optimum solution so far. In particular, here we choose the machine \(M6\) and insert it before the machine \(M2\), before the machine \(M1\) etc. and end with the insertion of it before machine \(M4\). Thus we have 6 candidate solutions – the one before inserting and for all 5 inserted possibilities. We choose the combination that gives us the best solution, in this case \((M2, M1, M3, M6, M5, M4, M7, M8, M9, M10)\) and continue by reinserting the next element – machine \(M7\). We insert machine \(M7\) into 6 possible positions, before machine \(M2, M1\) etc. and end with the insertion of it before \(M4\) and take the best solution which becomes the initial state before reinserting the machine \(M8\). We do this until we reach the last insertion of the last machine (in our case the machine \(M10\)). The best solution of this second step is \((M2, M1, M7, M3, M9, M10, M6, M5, M4, M8)\).

When we finish the step 2 we repeat the algorithm by using the best solution we got so far.

The algorithm repeats these two steps until it can no longer find an improvement.

After finding an arrangement for which no improvement was found by selection the first \(m\) machine, the procedure repeats by selecting the machine on positions \(2, \ldots, m+1, \ldots, n\).
iv. Computational results

The algorithm was tested at the possible realistic instance. We had 20 orders and 10 machines in production cell. We compared the algorithm with the genetic algorithm, which is already installed in the Siemens programming environment Plant Simulation [33]. The results are as follows:

Characteristics of the genetic algorithm (GA):
- 30 generations, size of generation: 100.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Total time</th>
<th>... for finding best solution</th>
<th>Quality of the best solution</th>
</tr>
</thead>
<tbody>
<tr>
<td>GA</td>
<td>1 min 52 s</td>
<td>43 s</td>
<td>€29593.5</td>
</tr>
<tr>
<td>RaR</td>
<td>1 min 1 s</td>
<td>15 s</td>
<td>€29593.5</td>
</tr>
</tbody>
</table>

From the results we see that RaR algorithm finds a very good solution in a relatively short time compared to GA. The great advantage of RaR algorithm is that there is not necessary to store large amounts of data, since the algorithm works sequentially, and in almost every step takes only best solution and discards the others.

In Figure 2 the Sankey chart is presented for the optimal arrangement of machines. Machines that have more of material flow are closer together so there are lower transport costs.

According to the tests that have been carried out (even those that are not described in here), we can say that the RaR algorithm works very well and reliably and in short time gives good solutions.

v. Conclusion

This paper proposes Remove and Reinsert (RaR) heuristics for the problem of arrangement of machines in the production cell. It minimizes the expected transport costs within a reasonable amount of calculation time.

For executions of different “what-if” scenarios of the initial schedules, the discrete event simulation (DES) software – Technomatix Plant Simulation was used. A comparison of RaR algorithm with Genetic Algorithm which is built-in module in Technomatix Plant Simulation software showed that RaR algorithm finds same optimal solution in shorter time compared to the Genetic Algorithm.

We also calculated the maximal transport cost for our case and the maximal transport cost we got was €43101.7 which represents 45.6% inferior combination compare to best solution.

In the real cell production the orders and amount of orders is changing, so this could be the effective tool for a cell planner to check once in a while if there is a need for the rearrangement of machines in the production cell.

In our future work, further experiments will be conducted to shorten the calculation time of getting the optimal solutions from the RaR algorithm. We will also consider different types of layouts in the production cell – U shape, star shape, square shape, etc.

**References**

Material flow modeling in system of parts. I could use genetic algorithms for the resource allocation problem in cellular manufacturing. My research focuses on developing algorithms to improve production cell layout generation and regeneration in multi-level supply systems. I have worked on an approximation algorithm for the generalized grouping problem in cellular manufacturing. My work has been published in reputable journals and conferences. I am a researcher at the Laboratory for Handling, Assembly and Pneumatics at the Faculty of Mechanical Engineering, University of Ljubljana, Slovenia. I wanted to use some algorithm for scheduling of machines in production cell. I could use genetic algorithms since they are very popular and they have almost God-like place in world of algorithms. But I asked myself if there is a way to create faster, simpler and more effective algorithm. Replace and Reinsert (RaR) algorithm that we developed has shown just that. If I quote from this paper: “... showed that RaR algorithm finds some optimal solution in shorter time compared to Genetic Algorithm.”