Int j simul model 16 (2017) 4, 644-657
ISSN 1726-4529 Original scientific paper

AN EFFECTIVE USE OF HYBRID METAHEURISTICS ALGORITHM FOR JOB SHOP SCHEDULING PROBLEM

Zhang, H.[*]; Liu, S.[*]; Moraca, S.[**] & Ojstersek, R.[***]
[*] School of Economics & Management, Beijing Jiaotong University, Beijing 100044, P. R. China
[**] Faculty of Technical Sciences, University of Novi Sad, Trg Dositeja Obradovića 6, 21000 Novi Sad, Serbia
[***] Faculty of Mechanical Engineering, University of Maribor, Smetanova 17, 2000 Maribor, Slovenia
E-Mail: zhanghankun@bjtu.edu.cn, shfliu@bjtu.edu.cn, moraca@uns.ac.rs, robert.ojstersek@um.si

Abstract
This paper presents an effective use of hybrid metaheuristics algorithm for solving Job Shop Scheduling Problem (JSSP). Integration of three metaheuristics algorithms: Shuffled Frog Leaping Algorithm (SFLA), Intelligent Water Drops algorithm (IWD) and Path Relinking (PR) algorithm were put together to solve JSSP. First, simulation model was developed and tested on the test data of Traveller Salesman Problem (TSP). Second, the model was tested on real world production line to solve the problem of Minimum Needed Workers (MNW) at the production line. The model enables individual test of three mentioned algorithms and calculation of new proposed Random Multi-Neighbourhood based Shuffled Frog Leaping Algorithm with Path Relinking (RMN-SFLA-PR). Experiments were tested on two software environments MATLAB and Simio, which gives us reliable, robust and tangible results. Results show that the new proposed RMN-SFLA-PR algorithm converged to optimum almost ten times faster than individual algorithms. The most important thing is the successful rate of all independent runs of the proposed RMN-SFLA-PR is 100 % in low-dimensional cases of the 4 benchmarks (dj38) and in JSSP to solve MNW for the real world production line.
(Received in April 2017, accepted in September 2017. This paper was with the authors 2 months for 1 revision.)

Key Words: Job Shop Scheduling Problem, Metaheuristics Algorithm, Shuffled Frog Leaping Algorithm, Path Relinking, Random Multi-Neighbourhood Structures

1. INTRODUCTION

The need of companies to have highly productive, reliable and economically efficient production line is a basic concept of Industry 4.0 [1]. In the past the main development of this three fields were made by human expertise in-depth knowledge, now scientists are more and more developing methods of artificial intelligence to solve different problems, in our case JSSP with MNW problem. Discrete systems simulation environments are used to simulate real world environments with real world parameters put into the simulation model. But now it is hard to solve NP-hard problems with just one individual simulation software environment. The main problem is creating new mathematical algorithm and putting it to the discrete system simulation environment, which already has some preprogrammed structures. That's why we proposed the exchange of data between two software environments: MATLAB, in which new mathematical algorithm is created, and discrete system simulation environment Simio, which is responsible for the real world production line testing.

The increase of production capacity is the main goal of modern production companies, but in most times the number of the workers must stay as low as possible in order to reduce the costs. That's why we propose a new RMN-SFLA-PR algorithm to calculate optimal MNW needed at the production line. The problem is presented in the paper first with the test data of TSP and second on real world production line, simulated in simulation software Simio.

Newer automated and robotized production line need workers mainly for checking and controlling the production line, which must be highly productive without any unplanned downtime. So, it is very important to properly allocate needed number of the workers to the

production line. We transfer the real world production data to simulation model including the following parameters: distances between the machine centres, utilization rate of machines, processing times of the machine centres, speeds of conveyor belts, workers and forklifts and time variables of the machine centres and worker's tasks, to define the MNW at the production line.

2. LITERATURE REVIEW

Industry 4.0 [1] is based on improving mass production in minimum needed time. The biggest world companies are striving for the maximum economic profit; they invest a lot of money to reach this goal. Without the implementation of artificial intelligence in the different science research fields this will not be possible or it will take a lot more time and money to solve these difficult problems.

Eusuff et al. [2, 3] presented the use of metaheuristic algorithm SFLA on the problem to determinate optimal discrete pipe size for new pipes networks and for the pipe network expansion. This was the first step of the implementation of the SFLA as particle swarm optimization (PSO) for discrete optimization. They proceed with developing next improved memetic metaheuristic algorithm in which the frogs act as carriers of memes where a meme is a unit of cultural evaluation. Seven years after this implementation of SFLA algorithm Xu et al. [4] use mansion algorithm for solving JSSP, NP-hard problem. They use hybrid encoding scheme in which two sequences are used to illustrate operations order, sequence and machine assignment. Along the improvement and the implementation of SFLA also PR algorithm was developed. Rahimi-Vahed et al. [5] present the routing problem of multi-depot periodic vehicle, for which they must minimize total travel costs. Ribeiro and Resende [6] review the fundaments and implementation strategies of PR algorithm. As well as numerical examples are discussed and algorithm are compared. After the basic individual research of metaheuristic algorithms, we can see that hybrid algorithms are arriving. Hybrid algorithms have a big advantage of combining positive think of individual algorithms and eliminating individual algorithms disadvantages. Nguyen et al. [7] proposed multi-star iterated local search (MS-ILS) to solve TSP the main depot and the customers. In this case, PR is used for reinforced the tabu list. In the paper Lai and Hao [8] use the PR algorithm for fixed spectrum frequency assignment problem. They use PR to generate intermediate paths and tabu search for local optimization. Multi-neighborhood based path relinking algorithm (MN-PR) for solving the two-sided assembly line balancing problem is proposed by Yang et al. [9]. The proposed MN-PR algorithm is able to improve the efficiency of the two-sided production line. PR algorithm is the basic method used for solving capacitated arc routing problem (CARP), the objective is to find a minimum cost set of tours servicing a subset of required edges under vehicle capacity constraints. The article is presented by Luiz Usberti et al. [10]. Chaves et al. [11] presented the minimization of minimum required tool switches to process a set of jobs on machine center. They use local search heuristics to simplify clustering process; the new hybrid heuristic based algorithm is named Biased Random Key Genetic Algorithm (BRKGA) and the clustering search (CS). After the implementing new metaheuristic algorithms to solve NP-hard problems, authors use simulations to optimize proposed models. Li et al. [12] present multi-objective optimization of cloud manufacturing service composition with cloud-entropy enhanced genetic algorithm which is very popular theme in the concept of Industry 4.0. Varga et al. [13] present gain-scheduling for hierarchical control. Nidhiry and Saravanan developed a modified non-dominated sorting genetic algorithm (NSGA-II) for multi-objective optimization and compared it with some existing algorithms [14]. Shah-Hosseini [15] in 2007 first introduced problem solving algorithm by IWD method. The basic mathematical model is presented in this paper, then two more papers are presented in 2008 and 2009 [16, 17] in

which multiple knapsack problem (MKP) and TSP are solved by IWD. In the area of JSSP we can see that a lot of work was done about integrating metaheuristic methods to solve thins NP-hard problem. Crawford et al. [18] introduced a proposal of design of Ant Colony Optimization algorithm paradigm using hyper-cube framework to solve the software project scheduling problem. PSO algorithm for finding Pareto-frontier in multi-objective JSSP is proposed by Wisittipanich and Kachitvichyanukul [19]. The objective of the paper is to simultaneously minimize makespan and total tardiness of jobs. The solutions found by the whole swarm are the new guidance for the particle movement. Sunder et al. [20] introduced a hybrid artificial bee colony algorithm (ABC) for JSSP with no-wait constraint; the proposed algorithm successfully coordinates initialization, selection and determination of the ABC with the local search which brings high quality solutions. Park et al. used multi-criteria optimization for the optimization of machining parameters [21]. Rodriguez-Tello et al. [22] investigated the role of evaluation function used by metaheuristics for solving combinatorial optimization problems.

3. SHUFFLED FROG-LEAPING ALGORITHM AND PATH RELINKING

3.1 Shuffled frog-leaping algorithm

The shuffled frog-leaping algorithm (SFLA) extends the shuffled complex evolution (SCE) algorithm and the particle swarm optimization (PSO) algorithm applicable to continuous optimization problems, it is a population-based and cooperative search metaphor inspired by natural memetics [2, 3]. The SFLA is a memetic meta-heuristic for solving discrete optimization problems [2, 3], e.g., the literature water distribution system problems [2], the groundwater model calibration problem [3] and the job-shop scheduling problem [4]. In the SFLA, a sample virtual population of frogs leaping in a swamp and searching for the optimum location of food [3]. During the memetic evolution, the frogs are infected by other better ideas and its memes are changed, resulting in a change in their position towards the goal [3].

3.2 Path relinking

The path relinking (PR) algorithm, as a population-based meta-heuristic and an intensification strategy to explore trajectories connecting elite solutions, which solves a given problem using purposeful and non-random exploration and exploitation strategies, is known as a powerful solution methodology [5, 6]. The PR was originally proposed to improve solutions obtained by tabu search or scatter search [5-7]. The PR is a major enhancement to heuristic search methods for solving combinatorial optimization problems, can be added to any metaheuristic algorithm, such as GRASP and genetic algorithms [6, 7]. Rahimi-Vahed et al. [5] propose an efficient path relinking algorithm whose exploration and exploitation strategies enable the algorithm to address the multi-depot periodic vehicle routing problem. Nguyen et al. [7] use the PR to reinforce the proposed multi-start iterated local search for the two-echelon location-routing problem. The main components of the general PR are the rules for building the reference set, the rules for choosing the initial and guiding solutions and a neighbourhood structure for moving along paths [5]. The PR can be classified into forward path relinking (the initial and guiding solutions are set to $S_i = x_2$ and $S_g = x_1$, respectively, where solution x_1 with better performance and solution x_2 with worse performance), backward path relinking (conversely, set to $S_i = x_1$ and $S_g = x_2$) and back-and-forward path relinking (backward path-relinking is applied first, followed by the forward path-relinking) [6]. As it moves along the path, the size of the restricted neighbourhood progressively decreases, the PR explores the neighbourhood of the initial solution more thoroughly than the guiding solution [6].

4. HYBRID METAHEURISTICS ALGORITHM WITH SFLA AND PR

4.1 Neighbourhood structures

The path relinking operator, as one of the most important components of the path relinking approach, aims to generate new promising solutions by creating paths connecting two high-quality parent solutions [8, 9]. Neighbourhood structure for moving along paths is the key for the path relinking operator. A neighborhood is typically defined as transforming a solution to generate an adjacent solution [9]. There are many commonly used neighbourhood structures, for example, insertion, two adjacent elements insertion, swap and two adjacent elements swap (see Fig. 1 a-d) [11]. Block-insertion, as the neighbourhood move used to generate the path, is easy to generate a series of moves [10]. Yang et al. [9] employs two complementary neighbourhood structures, insert move and swap move, can enhance the search effectiveness by combining them together. To transform from one solution to another in the search space, this paper employs two different neighbourhood structures, random block-insertion and random block-swap, where a random number of adjacent elements is set as:

$$b\_size = randi\big(min(j-i, d-j+1, Bmax)\big) \tag{1}$$

where i and j are two pre-selected positions, d is the dimension of decision variable for the practical problem, $Bmax$ is the maximum block size and $randi(.)$ is a function to generate a pseudorandom integer in the range $[1, .]$, . means parameter of the function.

The random block-insertion neighbourhood structure, denoted as NS_1, is defined by moving random number *b_size* adjacent elements from its original position to destination position (see Fig. 1 f). Similarly, the random block-swap neighbourhood structure, denoted as NS_2, is defined by exchanging a random number *b_size* of two different adjacent elements from its original position to the position of each other (see Fig. 1 g). The 2-opt, denoted as NS_3, as the most classical heuristic for the traveling salesman problem (TSP), is also considered as a basic neighbourhood structure in this paper. The 2-opt neighbourhood structure generates a new solution by reversing the order of the elements between two pre-selected positions (see Fig. 1 e).

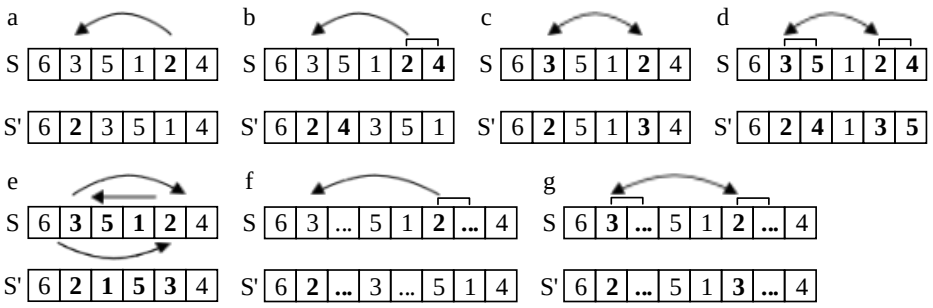

Figure 1: Examples of the neighbourhood structures.

4.2 Hybrid SFLA with PR

The PR, as advanced hybridizations with more elaborate metaheuristic schemes, can be applied to population-based stochastic local search algorithms as an advanced crossover or combination operator [6]. During the local search for each memeplex in the SFLA, which is frog-leaping, the best frog's position and worst frog's position in the submemeplex is determined once a submemeplex is constructed. They are used to generate a new position to improve the worst frog's position, meanwhile, its performance. If it cannot produce a better result, then the best frog's position in the submemeplex will be replaced by the best frog's position in the memeplex, to generate a new position to improve the worst frog's position. In

the two steps mentioned above, two positions are selected, which is also done during the PR operating, means that the PR can be applied to the local search for each memeplex in the SFLA as an advanced combination operator. In this paper, we present a random multi-neighbourhood based shuffled frog-leaping algorithm with path relinking (RMN-SFLA-PR) for solving combinatorial optimization problems. The applied order of the neighbourhood structures in [11] is fixed, which is determined in advance. This paper applies random neighbourhood structures size and randomly applied order of the neighbourhood structures. The general procedures of RMN-SFLA-PR and RMN-PR are shown in algorithms 1 and 2.

Algorithm 1 Pseudo-code of RMN-SFLA-PR

Step 0 Initialize parameters. Setting for the SFLA: $m, n, q, N, F = mn, d, CG, min\_ite, ite = 0$ and $CGCount = 0$. The neighbourhood structures $NS = \{NS_1, NS_2, NS_3\}$ and $Bmax$.
Step 1 Generate a virtual population. Generate F virtual frogs and compute their fitness f.
Step 2 Rank frogs. Sort the F frogs in decreasing order and record the best frog's position PX.
Step 3 Global search.
While $ite < min\_ite$ or ($CGCount < CG$ and $ite \geq min\_ite$)
  Step 4 Partition frogs into memeplexes.
  For $k = 1:m$
    for $j = 1:n$
      $Y(k,j) = F\big(k + m(j-1)\big), Yf(k,j) = f\big(k + m(j-1)\big)$
    end
  end
  Step 5 Shuffle memeplexes.
  For $im = 1:m$
    for $iN = 1:N$
      Step 5.1 The submemeplex Z is formed by randomly selected q distinct frogs. Record the best and the worst Wi frog's position as PB and PW respectively:
      $Wi = max(Z), PB = Y\big(im, min(Z)\big), PW = Y(im, Wi)$
      Step 5.2 Determine the neighbourhood structures size ns_size, which is applied in the current local search: $ns\_size = randi\big(size(NS)\big)$
      Step 5.3 Improve the worst frog's position by Algorithm 2.
      $[Pn, fn] = RMN - PR(PB, PW, Yf(im, Wi), ns\_size, NS, Bmax, d)$
      if $fn < Yf(im, Wi)$
        $Y(im, Wi) = Pn, Yf(im, Wi) = fn$
      else
        $[Pn, fn] = RMN - PR(PX, PW, Yf(im, Wi), ns\_size, NS, Bmax, d)$
        if $fn < Yf(im, Wi)$
          $Y(im, Wi) = Pn, Yf(im, Wi) = fn$
        else
          Randomly generate a new frog to replace the worst frog.
        End
      end
      Step 5.4 Upgrade the memeplex $Y(im)$ and sort in decreasing order.
    End
  end
  Step 6 Shuffle memeplexes. Replace memeplexes into the F frogs, sort in decreasing order and update the best frog's position PX. Record the consecutive count $CGCount$ and setting $ite = ite + 1$.
End

Algorithm 2 Pseudo-code of RMN-PR

Input: $PB, PW, f\_Wi, ns\_size, NS, Bmax$ and d. Output: a new position Pn and its performance fn.
Step 0 Choose the initial and guiding solutions. Since backward path-relinking tends to perform the best, the initial and guiding solutions are set: $Si = PB, Sg = PW, Pn = [\,]$ and $fn = f\_Wi$.
Step 1 Multi-neighbourhood based PR.

While $Si \neq Sg$

Step 2 Determine the applied order $od = randperm(size(NS))$ of the neighbourhood structures, where $randperm(.)$ generates a vector containing a random permutation of the integers $1: size(NS)$.

For $k = 1: ns\_size$

Step 3 The current neighbourhood structure is set as $ns = NS(od(k))$. Setting $x = Si$ and $i = 1$.

While $x \neq Sg$

if $x(i) = Sg(i)$

$i = i + 1$

else

for $j = i + 1: d$

if $x(j) = Sg(i)$

break;

end

end

Step 4 According to the neighbourhood structure $ns(x, i, j, d, Bmax)$, produce a new solution x and compute its performance fx.

If $fx < fn$

$Pn = x, fn = fx$

end

end

end

4.3 RMN-SFLA-PR test

In this paper, algorithms were implemented in MATLAB language. All experiments are simulated in MATLAB version R2016b on a laptop with x64-processor Intel(R) Core(TM)2 Duo CPU P8700 2.53 GHz and 4 GB RAM in the environment of Windows 10 Version 1607 OS. For each instance, algorithms are independently run for 30 times.

The TSP, as a most famous NP-hard problem in combinatorial optimization, is serviced as a benchmark problem to evaluate the RMN-SFLA-PR. We select dj38 [23], berlin52 [24], pr76 [24] and rd100 [24] as the benchmark instances. Their optimal tours have length 6659.43, 7544.37, 108159.44 and 7910.40, respectively. Fig. 2 shows the RMN-SFLA-PR solution convergence for the benchmark instances.

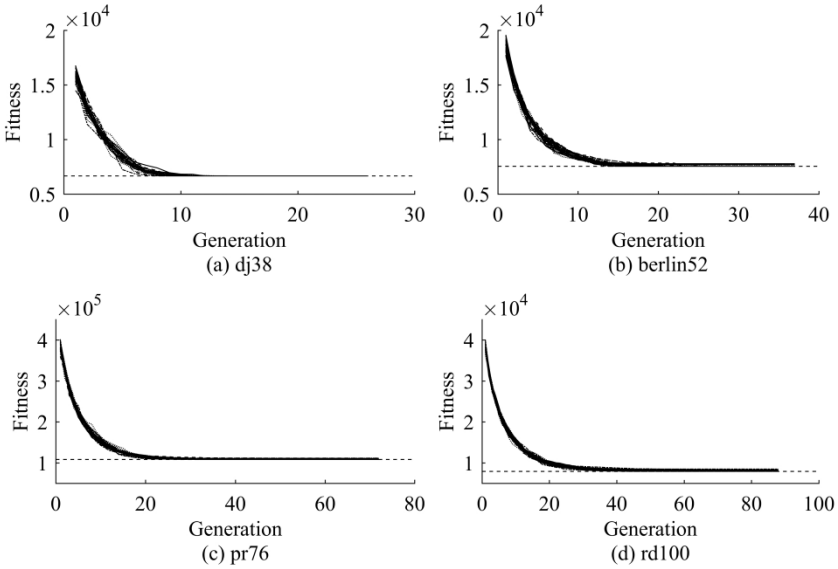

Figure 2: The RMN-SFLA-PR convergences of the best solutions for the benchmark instances (the horizontal dotted line is the length of the optimal tours for each instance).

The computational statistics of the fitness, the running time and the generation number of the 4 benchmark instances is shown in Table I and Table II, respectively. The proposed algorithm tends to converge to the best fitness in each benchmark instance. The successful rate of the all independent runs is 100 % in the dj38. And it can converge to global optimization in a very short time. However, as the dimensions of the benchmark instances increase, the success rate decreases, but meanwhile, the running time and the generation number increase. The robustness of the proposed algorithm decreases as the dimension of the problem increases.

Table I: Computational statistics of the RMN-SFLA-PR on the fitness for the 4 benchmark instances.

Name	**Min**	**Max**	**Mean**	**Standard deviation**	**Success rate** (%)
dj38	6659.43	6659.43	6659.43	0	100
berlin52	7544.37	7782.98	7583.79	74.68	76.67
pr76	108159.44	111375.83	109185.02	972.47	26.67
rd100	7910.40	8484.78	8117.39	140.25	3.33

Table II: Computational statistics of the RMN-SFLA-PR on the running time and the generation number for the 4 benchmark instances.

Name	**Running time** (s)				**Number of generation**			
	Min	**Max**	**Mean**	**Standard deviation**	**Min**	**Max**	**Mean**	**Standard deviation**
dj38	37.47	47.62	42.52	2.73	20	26	22.80	1.42
berlin52	56.55	83.63	67.65	6.69	26	37	30.27	2.94
pr76	122.58	220.13	146.26	21.32	39	72	48.93	7.01
rd100	225.08	373.52	270.52	36.61	49	88	62.47	8.80

5. COMPUTATIONAL EVALUATION

5.1 Solution coding

In this paper, the machines for the workers to check are represented by the index of the order in the production line. In order to encode the workers into the coding, we set the dimension of coding as $d = machineNum + workerNum - 1$, where *machineNum* is the number of machines in the problems, *workerNum* is the number of workers in the problems. It means that if the number of workers is more than 1, one or more decision variables, which are more than *machineNum*, are used as separators. For example, there are 8 machines for the workers to check, supposes we need two workers, a decision variable, its value is 9, is introduced into the coding. In Fig. 3, the decision variable 9 is serviced as a separator, can be decoded as worker 1, checks the machines and its sequence is (8, 1, 2, 6, 3) and worker 2 checks the machines and its sequence is (4, 5, 7).

S

8	1	2	6	3	9	4	5	7

Figure 3: Example of the solution coding.

5.2 Computational experiment

A manufacturing enterprise placed in Slovenia, denoted as C, has two automated production lines. They have the same machines, but the order of the machines on the production lines are the opposite. But they still need workers to check the machines in the automated production lines during their running. The number, name, position and checking time of the machines are shown in Table III.

Table III: Data for the production lines in the manufacturing enterprise C.

No	Name	Position (x, y)	Checking time (min)	No	Name	Position (x, y)	Checking time (min)
1	Robot 1	(0, 1.5)	15	9	Robot 1'	(3, 28)	15
2	Thread rolling	(0, 4.8)	17	10	Thread rolling'	(3, 26)	17
3	Induction slackening	(0, 9.1)	17	11	Induction slackening'	(3, 22)	17
4	Hard turning	(0, 13.5)	24	12	Hard turning'	(3, 16.5)	24
5	Hard milling	(0, 16.5)	55	13	Hard milling'	(3, 13.5)	55
6	Groove milling	(0, 22)	28	14	Groove milling'	(3, 9.1)	28
7	Groove controlling	(0, 26)	17	15	Groove controlling'	(3, 4.8)	17
8	Robot 2	(0, 28)	17	16	Robot 2'	(3, 1.5)	17

The time for the worker to walk from his current location of the machine *i* to his next location of the machine *j* is set as:

$$walking\_time(i,j) = \frac{distance(m_i, m_j)}{60 \cdot workerVel} \tag{2}$$

where *distance*(.,.)is a function to compute the Euclidean distance between two machines and *workerVel* represents the velocity of the workers walking which is set to 1.1 m/s.

There are *workerNum* workers in the C responsible for checking the production lines. The effective time for the worker *k* to work in one shift is set as:

$$working\_time(k) = \sum_{ki=1}^{kNum-1} walking\_time(ki, ki+1) + \sum_{kj=1}^{kNum} checking\_time(kj) \tag{3}$$

where *kNum* represents the number of the machines required for the worker *k* to check, *ki*, *ki* + 1 and *kj* represents the ki^{th}, $(ki+1)^{th}$ and kj^{th} machines in the checking sequence of worker *k*, respectively, and *checking_time*(*kj*) represents the time required for the worker to check the machine *kj*.

In the actual production process, each worker needs to complete a check in one shift. The effective working time of one shift in the C is 425 min, denoted as *shift_time*. So each worker's working time should be less than or equal to the *shift_time*. That is, the maximum worker's working time should be no more than the *shift_time*. The maximum worker's working time can be expressed as:

$$max\_working\_time = max(working\_time) \tag{4}$$

The constraint can be expressed as:

$$max\_working\_time \leq shift\_time \tag{5}$$

In this case, the number of workers, the number of machines for each worker to check and the checking sequence of the machines for each worker can affect the worker's effective working time, thereby affecting the maximum worker's working time. Therefore, at the pre-determined number of the workers, it is possible to optimize the number of machines for each worker to check and their checking sequence of the machines. The optimization objective function can be expressed as:

$$f = min(\max\_working\_time) \tag{6}$$

Once the optimal solution is obtained at the pre-determined number of the workers, denoted as *best_fitness*, the comparison can be made with the working time in one shift *shift_time*. If *best_fitness* ≤ *shift_time*, the pre-determined number of workers is sufficient. Otherwise, it is not enough. In order to determine the optimal number of workers, we can

reduce or increase the number of workers. We get the optimal number of workers when the following conditions are met:

$$\begin{cases} best\_fitness(workerNum) \leq shift\_time \\ best\_fitness(workerNum-1) > shift\_time \end{cases} \quad (7)$$

When the above conditions are met, the pre-determined number *workerNum* is the optimal number of workers.

5.3 Experimental results

For comparison, the JSSP was also solved by the SFLA and the intelligent water drops (IWD) algorithm. To evolve the solution, an operation similar to the modified precedence operation crossover, which is proposed by Xu et al. [4] in 2013, is applied to the local search of the SFLA. The IWD algorithm, as a swarm-based optimization algorithm, was inspired from observing natural water drops that flow in rivers for solving the TSP [15-17]. The IWD algorithm was first proposed by Shah-Hosseini in 2007 and then improved by him in 2008, denoted as IWD1 [15] and IWD2 [16], respectively. During the process of constructing a solution for each IWD in the IWD algorithm, the amount of soil on edges is used. It means that if the number of workers is more than 1, the soil of the edges between the one or more decision variables that are used as separators and the others are needed. The amount of soil between the two normal decision variables *i* and *j* in the JSSP is set as:

$$soil(i,j) = walking\_time(i,j) + checking\_time(j) \quad (8)$$

In the IWD algorithm, the probability of a path for being selected is inversely proportional to the amount of soil it has. The one or more virtual decision variables, which are used as separators, are preferred to be selected during the movement of the IWD for constructing a solution, rather than at the beginning or end. The amount of soil between the virtual decision variables and the others in the JSSP is set as:

$$soil(i,j) = mean(walking\_time) + mean(checking\_time) \quad (9)$$

where at least one of *i* and *j* is a virtual decision variable, the function *mean*(.) returns the mean value of its arguments.

According to [2, 3] and experiment, these parameter values are determined for the RMN-SFLA-PR: $m = 50$, $n = 15$, $q = 10$, $N = 20$, $CG = 10$, $min\_ite = 10$ and $Bmax = 3$. The parameter values of the SFLA are set the same as at the RMN-SFLA-PR, except for the *min_ite*, which is set to 300. Meanwhile, the parameter values of the IWD1 and IWD2 are set to the same as in [15] and [17], and the *CG* is also set the same as at the RMN-SFLA-PR while the *min_ite* is set to 3000.

The 4 algorithms' solution convergences for the JSSP with 1 and 2 workers are shown in Figs. 5 and 6, respectively. Fig. 7 shows the statistical analysis of the 4 algorithms for the JSSP with 1 and 2 workers. The computational statistics of the fitness, the running time and the generation number of the 4 algorithms, for the JSSP with 1 worker are shown in Table IV and Table V, and for the JSSP with 2 workers are shown in Table VI and Table VII, respectively. The RMN-SFLA-PR for the JSSP with 1 worker converges to 380.75 min. It means that 1 worker can finish checking all machines on the production lines in at least 380.75 min. For instance, the solution for the JSSP with 1 worker is shown in Fig. 4, S1. The RMN-SFLA-PR for the JSSP with 2 workers converges to 190.36 min. An instance of the solution for the JSSP with 2 workers is shown in Fig. 4, S2. That means, if there are 2 workers, they can complete in at least 190.36 min to check all the machines on the production lines. A worker can complete all the machines checks within the working time in one shift. So, only one worker in one shift is enough in the theory.

S1

7	8	9	10	11	6	5	12	13	4	3	14	15	2	1	16

S2

11	10	9	8	7	6	5	12	**17**	1	16	15	2	3	14	13	4

Figure 4: Instances for the solutions of the JSSP (the numbers represent the **no** in Table III and the number 17 is used as a separator in the S2).

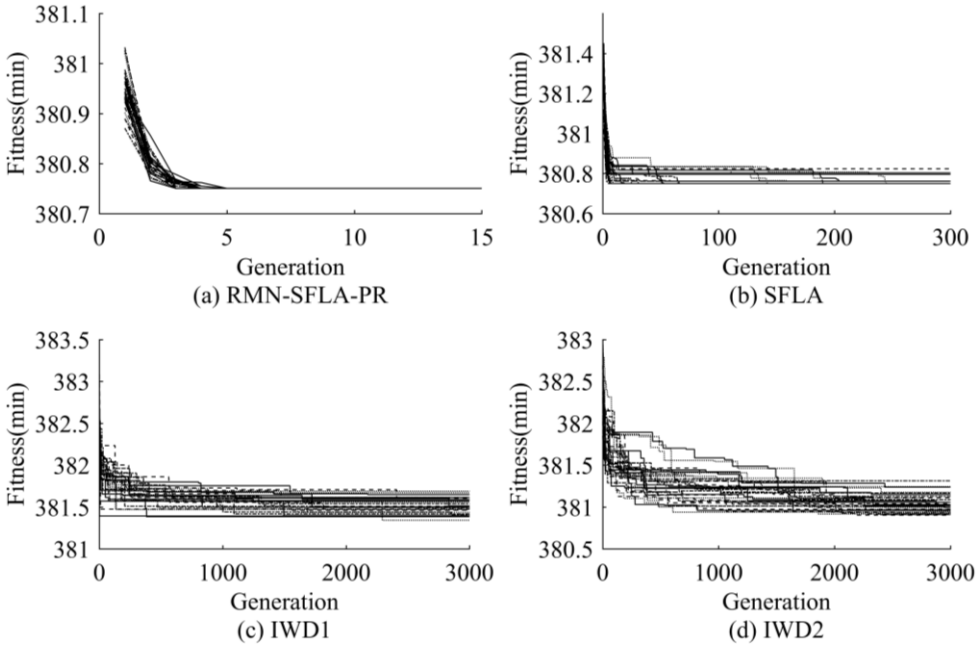

Figure 5: The 4 algorithms convergences of the best solutions for the JSSP with 1 worker.

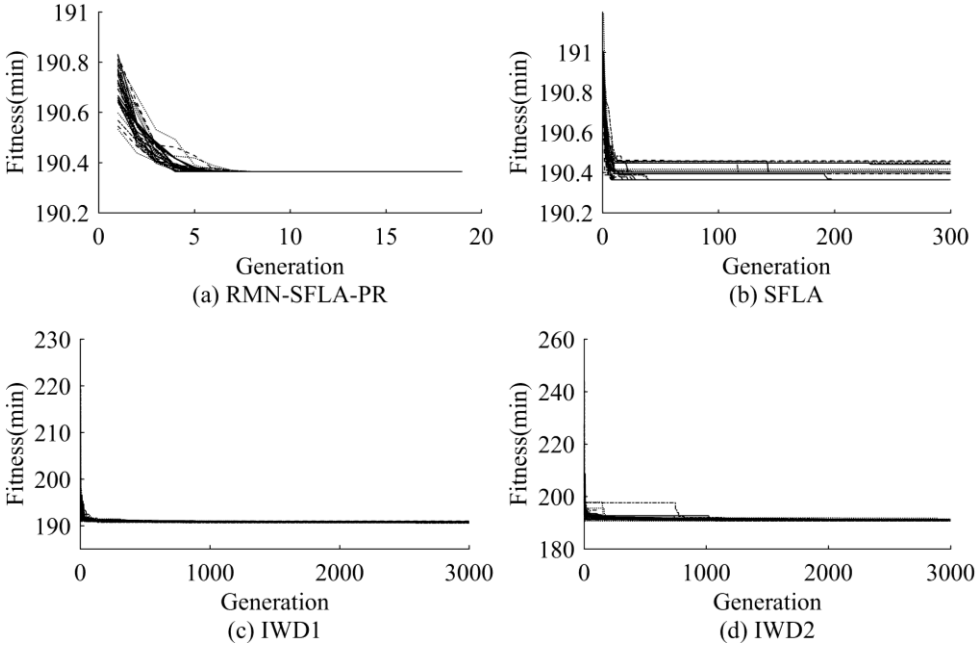

Figure 6: The 4 algorithms convergences of the best solutions for the JSSP with 2 workers.

The proposed algorithm tends to converge to the best fitness in both the JSSP with 1 and 2 workers. Its performance is the best in the 4 algorithms. The successful rate of all independent runs was 100 % in the proposed algorithm for both the JSSP with 1 and 2 workers. The performance of the SFLA is better, whose success rates for the JSSP with 1 and 2 workers is 46.67 % and 56.67 %, respectively. The IWD1 and the IWD2 perform the worst in both the JSSP with 1 and 2 workers, even cannot get the optimal solution. In the robustness of the

algorithms for both the JSSP with 1 and 2 workers, the best is also the proposed algorithm, followed by the SFLA, and the worst is the IWD1 and the IWD2. The robustness of the IWD1 for the JSSP with 1 worker is worse than the IWD2 but it is opposite for the JSSP with 2 workers. However, the IWD1 is more stable than the IWD2 in the robustness for both the JSSP with 1 and 2 workers. In the running time and the generation number for the JSSP with 1 and 2 workers, the algorithm presented in this paper is very time-friendly, but the other algorithms are more time consuming. The average running time of the proposed algorithm is almost one-tenth of the others. In both the JSSP with 1 and 2 workers, the average running times of the SFLA, the IWD1 and the IWD2 are almost the same. Although the average running time of the SFLA is more than both the IWD1 and the IWD2, it can obtain an optimal solution with almost 50 % probability in the all independent runs. The proposed algorithm is better than the others in both the JSSP with 1 and 2 workers.

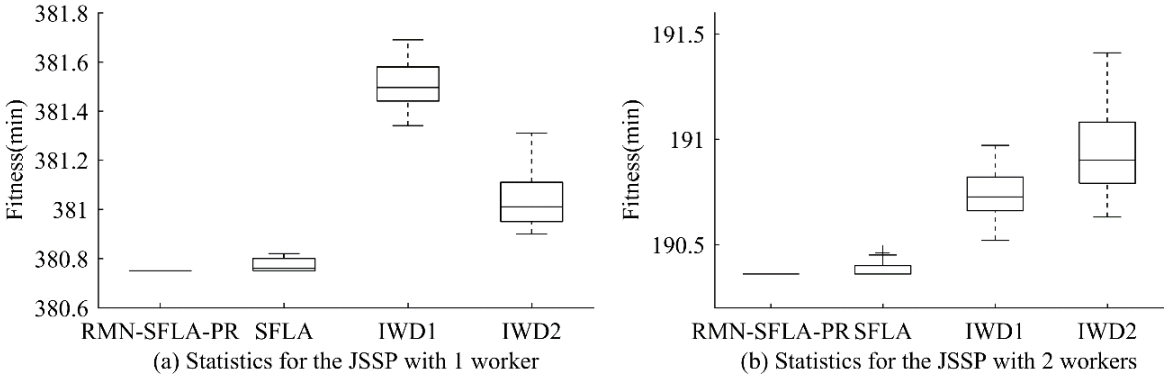

Figure 7: Statistical analysis of the 4 algorithms for the JSSP with 1 and 2 workers.

Table IV: Computational statistics of the 4 algorithms on the fitness for the JSSP with 1 worker.

Name	**Min**	**Max**	**Mean**	**Standard deviation**	**Success rate** (%)
RMN-SFLA-PR	380.75	380.75	380.75	0	100
SFLA	380.75	380.82	380.77	0.02	46.67
IWD1	381.34	381.69	381.51	0.09	0
IWD2	380.90	381.31	381.03	0.11	0

Table V: Computational statistics of the 4 algorithms on the running time and the generation number for the JSSP with 1 worker.

Name	**Running time** (s)				**Number of generation**			
	Min	**Max**	**Mean**	**Standard deviation**	**Min**	**Max**	**Mean**	**Standard deviation**
RMN-SFLA-PR	9.49	11.45	10.19	0.56	13	15	13.73	0.69
SFLA	116.36	133.73	120.42	3.84	300	300	300	0
IWD1	90.13	93.17	90.91	0.55	3000	3000	3000	0
IWD2	86.33	87.45	86.71	0.33	3000	3000	3000	0

Table VI: Computational statistics of the 4 algorithms on the fitness for the JSSP with 2 workers.

Name	**Min**	**Max**	**Mean**	**Standard deviation**	**Success rate** (%)
RMN-SFLA-PR	190.36	190.36	190.36	0	100
SFLA	190.36	190.46	190.38	0.03	56.67
IWD1	190.52	190.97	190.74	0.10	0
IWD2	190.63	191.41	190.94	0.19	0

Table VII: Computational statistics of the 4 algorithms on the running time and the generation number for the JSSP with 2 workers.

Name	Running time (s)				Number of generation			
	Min	Max	Mean	Standard deviation	Min	Max	Mean	Standard deviation
RMN-SFLA-PR	11.84	15.99	13.57	0.95	14	19	16.37	1.13
SFLA	121.50	129.52	123.82	1.81	300	300	300	0
IWD1	101.20	119.69	105.72	5.32	3000	3000	3000	0
IWD2	105.94	109.39	107.28	0.75	3000	3000	3000	0

6. SIMULATION

Our simulation model was created in discrete simulation environment Simio. Real life model of factory line C is presented in simulation model shown in Fig. 8. On the left of the figure, we can see the Simio 3D model of the production line. We put all necessary real world data from the production line to the simulation model: number of machine center (machine center parameters: utilization, number of produced peaces/h and time between failures), distances, times and number of workers which was calculated by RMN-SFLA-PR algorithm. On the right of the figure, we can see the 3D model of the production line drawn in the Autodesk Factory Design Suite.

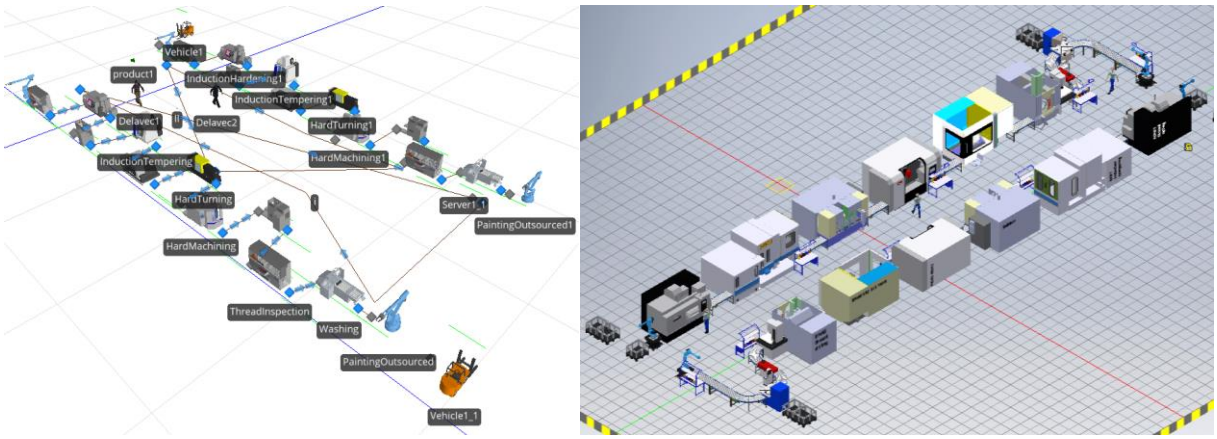

Figure 8: 3D models – in Simio (left) and in Autodesk Factory Design Suite (right).

In the theory, the C needs only one worker (calculated by RMN-SFLA-PR algorithm) to complete all the machines check in one shift. However, in the simulation of the Simio software, the performance of the machines is 85 %, that is, the effective working time of the machines (and workers) in one shift is set as:

$$machine\_time = shift\_time \cdot 85\ \% = 361.25\ \text{min} \tag{10}$$

The effective working time of the machines is 361.25 min in one shift. In this case, one worker is not enough to complete all the machines check in one shift. And taking into account the emergency situations that may occur during the production lines running, it is recommended to hire two workers to check the machines in one shift.

7. CONCLUSION

In this paper, we propose a random multi-neighbourhood based shuffled frog-leaping algorithm with path relinking for solving combinatorial optimization problems. The proposed RMN-SFLA-PR includes two different neighbourhood structures with random size block

operation, a random structure size and applied order of the multi-neighbourhood based local search strategy and a path relinking based local search guiding strategy.

We tested the proposed RMN-SFLA-PR on a set of 4 benchmark instances of the TSP. Computational results show that our algorithm is highly effective, especially in the cases of low-dimensional. For all instances, our proposed algorithm is always able to find the optimal or near optimal tours. Specifically, the successful rate of all independent runs is 100 % in the low-dimensional cases of the 4 benchmark instances. We apply the proposed RMN-SFLA-PR to solve the JSSP. The RMN-SFLA-PR performance was compared with the SFLA and the two IWD algorithms for the JSSP with 1 and 2 workers. The results for the applications show that the proposed RMN-SFLA-PR performed better than the others and was more robust in determining the global optimal solution and finding the solution speed. Followed by the SFLA, its performance is not as good as the proposed algorithm, but better than the two IWD algorithms, although it needs a relatively maximum running time. And both the IWD1 and the IWD2 performed poorly, whether its success rate or robustness.

Computational results show that one worker is enough to complete all the machines check in one shift for the C in the theory. However, we can see that the efficiency of the machine is only 85 %, tested by the Simio simulation. In addition, we should also need to consider the unexpected situations that may occur during the production lines running. Therefore, we recommend that the C employs two workers instead of one to check the machines in one shift.

One direction to extend this work is to more extensively test the proposed algorithm. Another is to improve the success rate of the proposed algorithm in the high-dimensional problem; such as improving the strategy for generating new solutions and the guiding strategy for local search, to increase the diversity of the proposed algorithm to escape from a local optimum.

ACKNOWLEDGEMENT

The study is supported by a project funded by the China Postdoctoral Science Foundation (043201003), by Beijing Natural Science Foundation (041501108), by National Natural Science Foundation (71132008, 71390334) and by Slovenian Research Agency (ARRS), founding No. P2-190.

REFERENCES

[1] Schwab, K. (2016). *The fourth industrial revolution*, Word Economic Forum, Geneva

[2] Eusuff, M. M.; Lansey, K. E. (2003). Optimization of water distribution network design using the shuffled frog leaping algorithm, *Journal of Water Resources Planning and Management*, Vol. 129, No. 3, 210-225, doi:10.1061/(ASCE)0733-9496(2003)129:3(210)

[3] Eusuff, M.; Lansey, K.; Pasha, F. (2006). Shuffled frog-leaping algorithm: a memetic meta-heuristic for discrete optimization, *Engineering Optimization*, Vol. 38, No. 2, 129-154, doi:10.1080/03052150500384759

[4] Xu, Y.; Wang, L.; Wang, S. (2013). An effective shuffled frog-leaping algorithm for the flexible job-shop scheduling problem, *Proceedings of the 2013 IEEE Symposium on Computational Intelligence in Control and Automation*, 128-134, doi:10.1109/CICA.2013.6611673

[5] Rahimi-Vahed, A.; Crainic, T. G.; Gendreau, M.; Rei, W. (2013). A path relinking algorithm for a multi-depot periodic vehicle routing problem, *Journal of Heuristics*, Vol. 19, No. 3, 497-524, doi:10.1007/s10732-013-9221-2

[6] Ribeiro, C. C.; Resende, M. G. C. (2012). Path-relinking intensification methods for stochastic local search algorithms, *Journal of Heuristics*, Vol. 18, No. 2, 193-214, doi:10.1007/s10732-011-9167-1

[7] Nguyen, V.-P.; Prins, C.; Prodhon, C. (2012). A multi-start iterated local search with tabu list and path relinking for the two-echelon location-routing problem, *Engineering Applications of Artificial Intelligence*, Vol. 25, No. 1, 56-71, doi:10.1016/j.engappai.2011.09.012

[8] Lai, X.; Hao, J.-K. (2015). Path relinking for the fixed spectrum frequency assignment problem, *Expert Systems with Applications*, Vol. 42, No. 10, 4755-4767, doi:10.1016/j.eswa.2015.01.025
[9] Yang, Z.; Zhang, G.; Zhu, H. (2016). Multi-neighborhood based path relinking for two-sided assembly line balancing problem, *Journal of Combinatorial Optimization*, Vol. 32, No. 2, 396-415, doi:10.1007/s10878-015-9959-6
[10] Luiz Usberti, F.; Morelato França, P.; Morelato França, A. L. (2013). GRASP with evolutionary path-relinking for the capacitated arc routing problem, *Computers & Operations Research*, Vol. 40, No. 12, 3206-3217, doi:10.1016/j.cor.2011.10.014
[11] Chaves, A. A.; Lorena, L. A. N.; Senne, E. L. F.; Resende, M. G. C. (2016). Hybrid method with CS and BRKGA applied to the minimization of tool switches problem, *Computers & Operations Research*, Vol. 67, 174-183, doi:10.1016/j.cor.2015.10.009
[12] Li, Y.; Yao, X.; Zhou, J. (2016). Multi-objective optimization of cloud manufacturing service composition with cloud-entropy enhanced genetic algorithm, *Strojniski vestnik – Journal of Mechanical Engineering*, Vol. 62, No. 10, 577-590, doi:10.5545/sv-jme.2016.3545
[13] Varga, B.; Nemeth, B.; Gaspar, P. (2015). Design of anti-roll bar systems based on hierarchical control, *Strojniski vestnik – Journal of Mechanical Engineering*, Vol. 61, No. 6, 374-382, doi:10.5545/sv-jme.2014.2224
[14] Nidhiry, N. M.; Saravanan, R. (2014). Scheduling optimization of a flexible manufacturing system using a modified NSGA-II algorithm, *Advances in Production Engineering & Management*, Vol. 9, No. 3, 139-151, doi:10.14743/apem2014.3.183
[15] Shah-Hosseini, H. (2007). Problem solving by intelligent water drops, *Proceedings of the IEEE Congress on Evolutionary Computation 2007*, 3226-3231, doi:10.1109/CEC.2007.4424885
[16] Shah-Hosseini, H. (2008). Intelligent water drops algorithm: A new optimization method for solving the multiple knapsack problem, *International Journal of Intelligent Computing and Cybernetics*, Vol. 1, No. 2, 193-212, doi:10.1108/17563780810874717
[17] Shah-Hosseini, H. (2009). The intelligent water drops algorithm: a nature-inspired swarm-based optimization algorithm, *International Journal of Bio-Inspired Computation*, Vol. 1, No. 1/2, 71-79, doi:10.1504/IJBIC.2009.022775
[18] Crawford, B.; Soto, R.; Johnson, F.; Misra, S.; Paredes, F.; Olguín, E. (2015). Software project scheduling using the hyper-cube ant colony optimization algorithm, *Technical Gazette*, Vol. 22, No. 5, 1171-1178, doi:10.17559/TV-20140519212813
[19] Wisittipanich, W.; Kachitvichyanukul, V. (2013). An efficient PSO algorithm for finding Pareto-frontier in multi-objective job shop scheduling problems, *Industrial Engineering and Management Systems*, Vol. 12, No. 2, 151-160, doi:10.7232/iems.2013.12.2.151
[20] Sundar, S.; Suganthan, P. N.; Jin, C. T.; Xiang, C. T.; Soon, C. C. (2017). A hybrid artificial bee colony algorithm for the job-shop scheduling problem with no-wait constraint, *Soft Computing*, Vol. 21, No. 5, 1193-1202, doi:10.1007/s00500-015-1852-9
[21] Park, H.-S.; Nguyen, T.-T.; Kim, J.-C. (2016). An energy efficient turning process for hardened material with multi-criteria optimization, *Transactions of FAMENA*, Vol. 40, No. 1, 1-14
[22] Rodriguez-Tello, E.; Hao, J.-K.; Romero-Monsivais, H. (2015). Boosting the performance of metaheuristics for the MinLA problem using a more discriminating evaluation function, *Technical Gazette*, Vol. 22, No. 1, 11-24, doi:10.17559/TV-20130905130612
[23] University of Waterloo. National Traveling Salesman Problems, from *http://www.math.uwaterloo.ca/tsp/world/countries.html*, accessed on 10-03-2017
[24] University of Heidelberg. TSPLIB95, from *http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/tsp/*, last modified on 06-08-2008, accessed on 01-02-2017