Pamukkale Üniversitesi Mühendislik Bilimleri Dergisi

Pamukkale University Journal of Engineering Sciences

Large neighbourhood search algorithm for type-II assembly line balancing problem

Tip-II montaj hattı dengeleme problemi için büyük komşuluk arama algoritması

Şener AKPINAR[1*]

[1]Department of Industrial Engineering, Engineering Faculty, Dokuz Eylul University, Izmir, Turkey.
sener.akpinar@deu.edu.tr

Received/Geliş Tarihi: 22.05.2016, Accepted/Kabul Tarihi: 22.08.2016 doi: 10.5505/pajes.2016.75975
* Corresponding author/Yazışılan Yazar Research Article/Araştırma Makalesi

Abstract

This paper proposes a large neighbourhood search (LNS) algorithm for type-II simple assembly line balancing problem (SALBP-II). The LNS algorithm was initially developed for solving vehicle routing problem and its later implementations were used to solve scheduling problems. The reported results about these two problems indicate that LNS algorithm is a powerful method. Vehicle routing problem has the main objective to find the optimal match between a certain number of routes and a certain number of customers, while SALBP-II is trying to find the optimal match between a certain number of workstations and a certain number of assembly operations. To our point of view, LNS algorithm would also be a powerful method for solving SALBP-II due to the structural similarity between these two problems. Within this context, a LNS algorithm is developed to tackle SALBP-II and the performance of the proposed algorithm is tested on a set of benchmark instances. Computational results indicate the satisfactory performance of LNS algorithm in solving SALBP-II.

Keywords: Large neighbourhood search algorithm, Assembly line balancing, Maximizing production rate, Meta-heuristic

Öz

Bu makale tip-II basit montaj hattı dengeleme problemi (BMHDP-II) için bir büyük komşuluk arama (BKA) algoritması önermektedir. BKA algoritması ilk olarak araç rotalama problemlerinin çözümü için önerilmiş ve sonraki uygulamaları çizelgeleme problemlerinin çözümü üzerine olmuştur. Bu iki problem için raporlanan sonuçlar BKA algoritmasının güçlü bir yöntem olduğunu ortaya koymuştur. Araç rotalama problemi belirli sayıdaki rota ile belirli sayıdaki müşteriler arasındaki optimum eşleşmeyi bulmak temel amacına sahipken, BMHDP-II belirli sayıdaki istasyon ile belirli sayıdaki montaj işlemleri arasındaki optimum eşleşmeyi bulmaya çalışmaktadır. Bizim açımızdan, bu iki problem arasındaki bu yapısal benzerlik BKA algoritmasının BMHDP-II için de güçlü bir yöntem olabileceği fikrini doğurmuştur. Bu kapsamda, BMHDP-II için bir BKA algoritması geliştirilmiş ve geliştirilen algoritmanın performansı bir problem seti üzerinde test edilmiştir. Hesaplamalı sonuçlar BMHDP-II çözümünde BKA algoritmasının tatmin edici performansını ortaya koymaktadır.

Anahtar kelimeler: Büyük komşuluk arama algoritması, Montaj hattı dengeleme, Üretim oranı maksimizasyonu, Meta-sezgisel

1 Introduction

The history of the assembly lines begins with the Terracotta Army, which was produced in an assembly type manner in China in 200 BC, however, assembly lines played their vital role in automotive industry after the development of the first automotive assembly line by Ransom Olds in 1901 [1]. In 1913, Henry Ford improved Ransom Olds' idea by introducing the moving belt for the assembly lines of his factory, where he had been produced Model T Fords. This innovative improvement on the automobile production process changed the type of manufacturing systems all over the world over the years because of the significantly reduced cost of production through the investment and installation of assembly lines (ALs). Through these kind of production systems companies have been able to produce standardized products in high volumes. Additionally, this type of production systems picked out a new optimization problem namely assembly line balancing problem (ALBP) that mainly deals with the design issues of ALs. ALBP searches for the allocation of task to workstations under some constraints in order to optimize some pre-determined performance measures.

Salveson formulated ALBP mathematically for the first time in literature and focused on line configuration, assignment of tasks to workstations [2]. Afterwards, ALBP has been attracted a lot of attention in the research community. This interest of the research community on the ALBP has been increased continuously. The reader can be informed about the literature on ALBPs through the comprehensive review studies of [3]-[7].

This paper considers the single-model assembly lines, designed to produce high volumes of standardized products, and simple assembly line balancing problem (SALBP) [8] tries to calibrate such lines. In the relevant literature, SALBP has been tackled as SALBP-I and SALBP-II in order to give some decisions concerning the design and operational levels, respectively. SALBP-II (respectively SALBP-I) is related to assignment of assemble tasks to workstations in order to identify minimum cycle time (respectively minimum number of workstations) for a pre-determined number of workstations (respectively cycle time) while satisfying the problem constraints. Both SALBP-I and SALBP-II are NP hard [9], and SALBP-I is much more popular among the researchers than the SALBP-II [7]. The reader can be suggested to see the review paper of [8] for detailed discussions about the solution approaches of simple assembly line balancing problems.

Solution approaches for SLABP-II can be evaluated as direct and iterative solution strategies. A direct solution method aims at identifying the optimum solution directly, while an iterative solution approach tries to identify the optimum solution by tackling SALBP-I for different trial cycle times.

Additionally, solution methods on SALBP-II can be grouped as exact and heuristic approaches. The exact methods developed in [10] and [9] are able to solve the SALBP-II iteratively, while the one developed in [11] is able solve the problem directly. Moreover, integer-programming formulations for SALBP-II were developed in [12] and [13]. Besides these exact methods, the SALBP-II literature covers many heuristic and meta-heuristic approaches. A linear programming based two-stage heuristic was proposed in [14] while a Petri nets based heuristic was proposed in [15]. Another heuristic method that starts an initial solution and realizes trade, and transfer procedures to improve this solution was proposed in [16]. Genetic algorithms [17]-[19], tabu search procedures [20],[21], simulated annealing algorithms [22],[23], ant colony optimization algorithms [24],[25], particle swarm optimization algorithm [26], differential evolution algorithm [27],[28], beam search algorithm [28]-[30] and variable neighbourhood search [31],[32] were also used to solve SALBP-II. In this paper, we propose a large neighbourhood search (LNS) algorithm for solving the SALBP-II. The LNS algorithm and its variants have been successfully implemented to solve vehicle routing problems in literature (see Section 2). Vehicle routing problem and the SALBP-II have structural similarities as the vehicle routing problem is trying to optimally match a certain number of routes and a certain number of customers, while SALBP-II is trying to optimally match a certain number of workstations and a certain number of assembly operations. Therefore, LNS algorithm would also be a powerful method for solving SALBP-II due to the structural similarity between these two problems. To the best of our knowledge, this is the first attempt to solve the SALBP-II via the LNS algorithm, however a simulation-based adaptive large neighborhood search heuristic was developed in [33] for a specific case of assembly line balancing problem, designing of a footwear assembly line under stochastic task time and parallel workstations.

The remainder of this paper is organized as follows. Basic steps of the LNS algorithm are given in Section 2. The proposed LNS algorithm for solving SALBP-II is presented in Section 3. Computational study is presented in Section 4. Conclusions are given in Section 5.

2 Large neighbourhood search algorithm

Nomenclature

s	A feasible solution,
s'	Solution under improvement,
s_{best}	Best solution,
$f(s')$	Fitness value of the solution s',
$f(s_{best})$	Fitness value of the best solution,
q and p	User defined parameters,
L	An array sorting individual effects of tasks on a solution,
y	A random number from the interval $[0,1)$.
r	Task to be removed from a solution

Shaw [34] initially proposed the LNS algorithm for solving vehicle routing problem with time windows (VRPTW). The LNS algorithm tries to explore the solution space of an optimization problem through two successive operators: destruction of a solution by a removal heuristic and reparation of the destroyed solution by an insertion heuristic. A removal heuristic generates an infeasible solution by removing some components of a solution, while an insertion heuristic turns this infeasible solution to a feasible solution by reinserting the removed components with the guidance of a rule. Then, LNS algorithm accepts or declines this solution via an acceptance function. LNS algorithm executes these steps successively as can be seen from Figure 1 until the stopping condition is met.

```
Initialize $s \in \{solutions\}$ and $q \in \mathbb{N}$
  $s_{best} \leftarrow s$
    repeat
        $s' \leftarrow s$
        project $q$ components out of $s'$
        reinsert removed components into $s'$
        if $(f(s') < f(s_{best}))$ then
            $s_{best} \leftarrow s'$
        if accept$(s', s)$ then
            $s \leftarrow s'$
    until termination condition met
report $s_{best}$
```

Figure 1: Main steps of the LNS algorithm [35].

LNS algorithm and its variants are powerful methods for routing and scheduling problems in general [36]. Ropke and Pisinger modified the basic LNS algorithm and they named the modified algorithm as adaptive large neighbourhood search (ALNS) algorithm [35]. ALNS has many successful implementations in literature [37]-[44]. Within the scope of this current paper, we have the aim of tackling the SALBP-II via a LNS algorithm and this attempt will be the first one for solving an ALBP via a LNS algorithm in literature as far as we know.

3 A large neighbourhood search algorithm for SALBP-II

This section introduces the proposed LNS algorithm for solving SALBP-II in depth. The proposed algorithm initializes itself by randomly generating a feasible solution and tries to improve this solution through removal and insertion heuristics. The algorithm terminates itself when the stopping condition is achieved. The following sub-sections introduce the main steps of the proposed LNS algorithm for SALBP-II.

3.1 Solution representation

In the proposed LNS algorithm, we used task based representation [45],[46], which is used for type-I ALBPs in general. The number of tasks defines the length of the representation schema; however, the situation is different for type-II problems. A solution representation schema of type-II ALBPs must contain the information about the pre-determined number of workstations. Because of this reason, the original task based representation must be modified to code a solution of the SALBP-II. Within this context, we modified the original task based representation as illustrated in Figure 2.

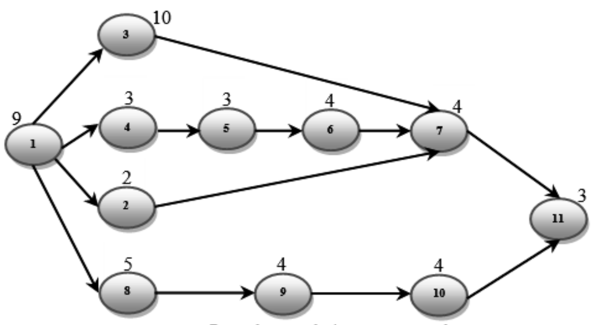
1	2	5	0	4	6	7	3	0	8	9	0	10	11

Figure 2: Solution representation for SALBP-II.

Figure 2 represents a SALBP-II solution of an assembly line having 11 tasks and these tasks must be performed within four workstations. In addition to the original task based representation, the modified version contains separators as “0” for determining tasks assignments to a pre-defined number of workstations. As can be seen from Figure 2, three separators used for determining tasks assignments to four workstations. Thus, the representation schema for 11 tasks and 4 workstations problem has 14 digits. According to this representation, tasks 1, 2 and 5 are performed in workstation 1, tasks 4, 6, 7 and 3 are performed in workstation 2, tasks 8 and 9 are performed in workstation 3, and finally tasks 10 and 11 are performed in workstation 4. The workload of each workstation is calculated as the summation of task times assigned to the related workstation and the maximum workload specifies the occurred cycle time on the line according to the related solution.

3.2 Initial solution

As mentioned before, the proposed LNS algorithm starts with a randomly generated initial feasible solution and tries to improve this solution through its operators. For the SALBP-II, the initial solution generation mechanism firstly determines a task sequence that satisfies the precedence relations between tasks. After that, it divides the related task sequence into *s* number of workstations by inserting *s*-1 separators into randomly selected positions within the task sequence. Figure 3 illustrates the initial solution generation mechanism for an assembly line having 11 tasks and 3 workstations.



a- Precedence relations among tasks

1	4	8	5	3	2	9	6	10	7	11

b- A feasible task sequence according to precedence relations

1	4	8	5	0	3	2	9	0	6	10	7	11

c- A feasible solution representation for SALBP-II

Figure 3: Initial solution generation mechanism.

The precedence relations and the task performing times are shown in Figure 3a and the Figure 3b illustrates a feasible task sequence according to precedence relations among tasks. For determining tasks assignments to three workstations 2 separators are inserted into the randomly determined positions within the feasible task sequence as can be seen from Figure 3c. According to the randomly generated solution in Figure 3c, tasks 1, 4, 8 and 5 are performed in workstation 1, tasks 3, 2 and 9 are performed in workstation 2, and tasks 6, 10, 7 and 11 are performed in workstation 3. These assignments of tasks result to workloads for workstations 1, 2 and 3 as 20, 10 and 15, separately. As a result, the cycle time of the assembly line is determined as 20 due to the solution depicted in Figure 3.

3.3 Removal heuristic

The proposed LNS algorithm realizes a removal heuristic based on the worst removal heuristic [35] in order to destroy the current solution. This main idea of this heuristic is specifying the misplaced tasks within a solution and inserting them into another position in the current solution. Within this scope, the proposed removal heuristic calculates a cost value of performing task i in its current position in the current solution as $cost(i,s) = f(s) - f_{-i}(s)$ where $f(s)$ is the cost of the complete solution and $f_{-i}(s)$ is the cost of the solution without task i. For the SALBP-II implementation of the worst removal heuristic, we considered the total idle time occurring on the assembly line. Hereby, the proposed removal heuristic aims at identifying the tasks causing high idle times within the assembly line according to current solution. In other words, the proposed removal heuristic aims at specifying the tasks inducing high cycle time in its current position in the current solution. The algorithm of the worst removal heuristic is depicted in Figure 4.

```
Inputs: $s \in \{solutions\}$, $q \in \mathbb{N}$, $p \in \mathbb{R}_+$
 while $q > 0$ do
   Array: $L$= Sorted tasks by decreasing $cost(i,s)$
   generate $y$ randomly from the interval $[0,1)$
       $r \leftarrow L[y^p|L|]$
       remove task $r$ from solution $s$
       $q \leftarrow q - 1$
 end while
```

Figure 4. Worst removal heuristic

As can be seen from Figure 4, removal heuristic uses two parameters as q and p for determining the number of tasks to be removed and avoiding repeatedly removing same tasks, respectively. As mentioned before, this heuristic tries to specify the misplaced tasks in a solution and therefore identifies the tasks required to move to another position in the current solution. Various removal heuristics were developed especially for the adaptive versions of the LNS algorithm, but there is no information about the best removal heuristic in literature. Therefore, we select the more reasonable removal heuristic for us, since our focus is to implement the LNS algorithm for SALBP-II not identifying the best removal heuristic within the context of this paper.

3.4 Insertion heuristic

In this paper, we used an insertion heuristic based on the ranked positional weight technique (RPWT) [47], which calculates positional weights for all the tasks of an assembly line according to the precedence relations and task performing times. The positional weights are then used as priority values while assigning tasks to workstations. For the SALBP-II implementation of the LNS algorithm, it is required to determine the order of reinserting the removed tasks by the removal heuristic. Within this scope, the insertion heuristic used in this paper firstly ranks the removed tasks according to their positional weights in descending order. After that, the insertion heuristic tries to reinsert the removed tasks in its best position in accordance with their ranks. Various insertion heuristics because of different priority values can be used for the SALBP-II implementation of the LNS algorithm, however we mainly focus on the implementation of LNS algorithm to SALBP-II not selecting the best insertion heuristic within the context of this study.

3.5 Solution acceptance and stopping criteria

The original LNS algorithm accepts only the improving solutions; however, this may cause to be trapped in a local optimum. Therefore, the later variants of LNS algorithm realize an acceptance function in order to specify the current solution. In our implementation of LNS algorithm to SALBP-II, we use an acceptance criteria based on the simulated annealing algorithm's acceptance criteria as realized in [35]. That is, the proposed LNS algorithm accepts a solution sol' in comparison to the current solution sol with the probability of $e^{-(f(sol')-f(sol))/T}$ where $T > 0$ defines the current iteration's *temperature*. The temperature has an initial value as T_{start} and it decreases via the rule $T = T.cr$, where $0 < cr < 1$ is the cooling ratio. The proposed LNS algorithm terminates itself after executing a pre-determined number of iterations or achieving the previously known optimal solution of a problem.

4 Computational study

This section evaluates the performance of the proposed LNS algorithm. For this purpose, we used the benchmark data sets downloaded from the website of http://alb.mansci.de/. The benchmark sets contain 302 instances in total. 128 instances based on 9 different precedence graphs belong to the data set 1 with the number of tasks varying from 29 to 111 and the rest of the 174 instances based on 8 different precedence graphs belong to the data set 2 with the number of tasks varying from 53 to 297. Each SALBP-II instance is defined with the precedence relations between tasks illustrated via a precedence graph G and the processing times of each task. Moreover, n number of tasks must be assigned to a pre-determined number of m workstations in order to identify the existing minimal cycle time c^*.

4.1 Results and Discussion

In this sub-section, we have directly compared the proposed LNS algorithm to 15 formerly developed SALBP-II methods, 3 versions of genetic algorithm (pGA [48], rGA_prb and rGA_rks [49]), 4 versions of differential evaluation algorithm (DE-prb and DE_rks [27], IDEA_rd2 and IDEA_opt [28]), 3 petri net based heuristics (PNA_for, PNA_back and PNA_bid [15]), 4 versions of particle swarm optimization (PSO1, PSO2, PSO3 and PSO4 [26]) and iterated beam search (IBS [29]). The proposed LNS algorithm was coded in Matlab 7.9 and then the results were obtained by running the coded LNS on a Core(TM) i7-2640 CPU 2.80 GHz personal computer. Additionally, LNS requires tuning some parameters as summarized in Table 1 with their values and definitions.

The values of the parameters were determined through a preliminary experimental study executed on the problem of Killbridge with 45 tasks and 11 workstations. This preliminary study was performed with three levels of $MaxIter \in \{50, 100, 150\}$, three levels of the lower limit (LL) and upper limit (UL) for the rule that randomly selects the number of tasks to be removed as $LL \in \{4, 8, 12\}$ and $UL \in \{\min(50, 0.2n), \min(50, 0.4n), \min(50, 0.6n)\}$, and three levels $p \in \{4, 6, 8\}$. The values for the parameters T_{start} and cr were taken from [50]. This experimental study done for parameter tuning resulted with the parameter values given in Table 1 by considering the quality of the generated solutions and computational time spent by the algorithm.

Table 1: Parameters of the proposed LNS.

Parameter	Definition	Value
MaxIter	Maximum number of iterations	100*n
q	Number of tasks to be removed	$4 \leq randi() \leq \min(50, 0.4n)$
p	Parameter used to avoid repeatedly selecting same task for removal	6
T_{start}	Initial temperature	100
cr	Cooling ratio	0.99975

Through this preliminary study we can provide following observations. The higher values of $MaxIter$ caused redundant iterations while its smaller values result with disappointed performance of the algorithm. Higher values of the LL of the random selection rule caused unsatisfactorily level of intensification., while the lower values of the UL of the random selection rule caused unsatisfactorily level of diversification. Finally, lower values of p cooresponds to much randomness as stated in [35]. The results obtained via the proposed LNS algorithm and the provided results of the other 15 algorithms on the benchmark sets are reported in Table 2. We ran the proposed LNS algorithm 10 times for all the test problems, since the reported results for the majority of other methods had also been obtained through 10 independent runs of algorithms for each problem. Table 2 provides the following information for all the methods.

- **c%dev:** The average relative deviation from the known optimum solution (previously known minimal cycle time c^*) in percentage; calculated as $\left((c - c^*)/c^*\right)100$; where c is the best cycle time achieved by the solution algorithm.

From the observations of Table 2, it can be clearly seen that the proposed LNS algorithm outperformed the variants of genetic algorithm, differential evaluation algorithm, particle swarm optimization and petri net based heuristics with respect to the c%dev achievements, which is related to the main optimization criteria of cycle time. However, IBS outperforms all the other algorithms, as well as LNS as can be obviously seen from Table 2. The results of all the algorithms are visualized in Figure 5, which displays the fluctuations of c%dev values over the benchmark problems, in order to emphasize the effective performance of the proposed LNS algorithm in solving SALBP-II instances.

The effectiveness of the LNS algorithm in solving the SALBP-II instances can be clearly seen if the average %cdev values provided for the data sets seperately taken into consideration.

LNS was able to generate solutions for data set 1 and data set 2 with an average %cdev value of 0.78 and 1.02, respectively. Therefore, we can conclude that LNS is a powerful method for solving SALBP-II instances. Finally, average %cdev values in comparison to other methods are visualized in Figure 6 in order to provide a better undestanding.

5 Conclusions

The main idea of this study was to tackle SALBP-II with a LNS algorithm for the first time in literature as far as we know. The original LNS algorithm and its later variants were generally used to tackle vehicle routing and scheduling problems. Especially for the vehicle routing problems, variants of the LNS algorithm is able to produce satisfactory results. Due to the similar nature of vehicle routing and type-II ALBP, it was thought that the LNS algorithm would be a satisfactory method in solving type-II ALBPs. Within this context, a LNS algorithm was developed to tackle a well-known assembly line balancing problem of SALBP-II.

Table 2: Computational results over the benchmark set.

Problem Name	n	pGA	rGA_ rks	rGA_ prb	DE_ prb	DE _ rks	IDEA _rd2	IDEA _apt	PNA_ for	PNA_ back	PNA_ bid	PSO1	PSO2	PSO3	PSO4	IBS	LNS
Data Set 1																	
Buxey	29	3.26	2.15	2.46	2.69	1.16	2.21	3.07	3.07	2.92	3.76	1.96	5.88	1.55	5.01	0	0.55
Sawyer	30	2.61	4.51	3.67	3.52	2.27	3.34	3.52	4.00	4.39	5.24	2.24	6.62	1.60	5.61	0	1.22
Lutz1	32	0.32	0.71	0	0.35	0.32	0.35	0.71	4.09	2.93	4.09	-	-	-	-	0	0.32
Gunther	35	0.64	1.71	1.21	1.02	0.14	1.02	1.27	1.27	2.99	1.27	0.87	3.88	0.64	3.92	0	0.42
Kilbridge	45	1.38	0.96	0.8	0.6	0.66	0.60	0.60	2.19	1.46	1.91	0.80	1.64	0.68	1.78	0	0.07
Tonge	70	2.75	3.63	3.63	2.07	1.88	2.07	1.78	2.53	3.41	2.92	2.91	5.99	2.03	5.82	0	1.03
Arcus1	83	1.65	1.96	1.48	0.83	0.99	0.98	0.78	2.47	2.57	2.25	1.42	3.20	0.70	3.04	0.0287	0.70
Lutz2	89	3.13	3.76	2.84	2.54	3.08	2.64	1.99	2.99	4.44	3.92	-	-	-	-	0	0.85
Arcus2	111	5.02	5.27	5.11	4.98	4.96	4.98	4.64	2.06	4.61	2.25	3.79	6.67	2.02	5.77	0.0066	1.82
Average		2.31	2.74	2.36	2.07	1.72	2.02	2.04	2.57	3.51	2.88	2.00	4.84	1.32	4.42	0.0058	0.78
Data Set 2																	
Hahn	53	0	1.27	0	0	0	0	0	2.52	2.08	2.61	-	-	-	-	0	0.50
Warnecke	58	4.29	6.40	5.25	3.31	3.74	3.98	3.51	5.57	5.90	6.01	-	-	-	-	0.0579	0.35
Wee-Mag	75	2.61	2.37	2.44	1.39	1.23	1.63	1.39	1.56	1.63	1.57	-	-	-	-	0	0
Lutz3	89	2.54	3.20	2.19	1.58	1.68	2.88	1.57	2.59	2.88	2.94	-	-	-	-	0	0.78
Mukherje	94	-	-	-	-	-	-	-	1.04	1.41	1.21	-	-	-	-	0	0.64
Barthold1	148	0.85	0.76	0.58	0.24	0.26	0.46	0.26	1.02	0.46	0.48	-	-	-	-	0	0.64
Barthold2	148	9.64	9.77	9.88	7.37	6.85	4.79	4.79	3.97	4.79	4.32	-	-	-	-	0	2.14
Scholl	297	10.16	11.76	10.31	9.87	9.51	10.16	9.24	3.21	2.90	2.75	-	-	-	-	0.0028	3.13
Average		4.27	5.08	4.38	3.39	3.32	3.41	2.97	2.85	3.01	2.93	-	-	-	-	0.00071	1.02

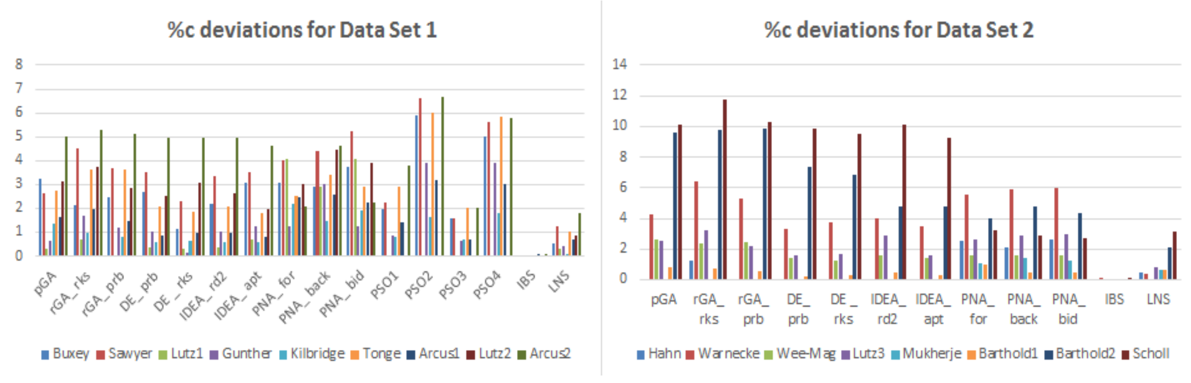


Figure 5: The average relative deviations over the benchmark instances.

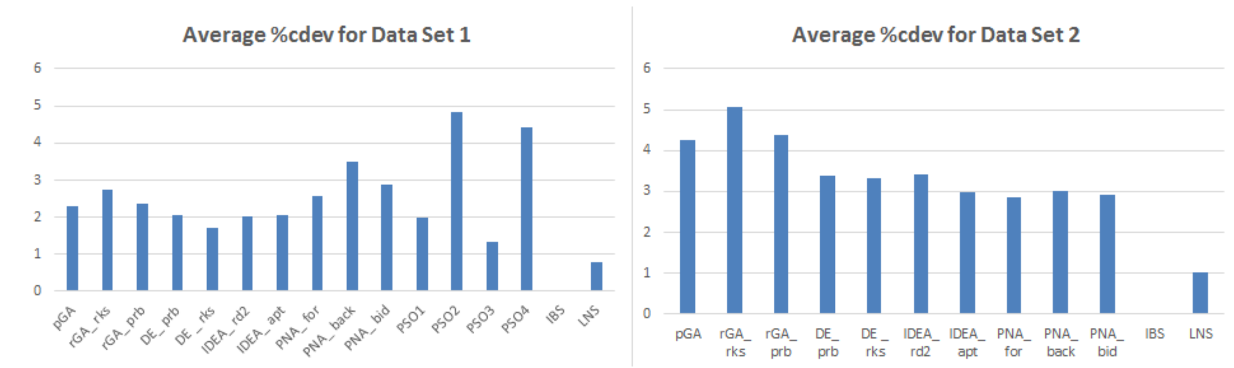


Figure 6: Average %cdev values according to different algorithms.

The performance of the proposed LNS algorithm was evaluated on a set of SALBP-II instances against 15 formerly developed SALBP-II methods, 3 versions of genetic algorithm, 4 versions of differential evaluation algorithm, 3 petri net based heuristics, 4 versions of particle swarm optimization and iterated beam search. Computational results demonstrate that the proposed LNS algorithm is capable to SALBP-II instances with a satisfactory performance.

Future researches will focus on tackling the different versions of ALBP via the proposed LNS algorithm and developing some other variants of the LNS algorithm for the ALBPs in order to improve the effectiveness of the proposed LNS algorithm.

6 References

[1] Thomopoulos NT. *Assembly Line Planning and Control*. Switzerland, Springer, 2014.

[2] Salveson ME. "The assembly line balancing problem". *Journal of Industrial Engineering*, 6, 18-25, 1955.

[3] Ghosh S, Gagnon RJ. "A comprehensive literature review and analysis of the design, balancing and scheduling of assembly systems". *The International Journal of Production Research*, 27(4), 637-670, 1989.

[4] Erel E, Sarin SC. "A survey of the assembly line balancing procedures". *Production Planning & Control*, 9(5), 414-434, 1998.

[5] Becker C, Scholl A. "A survey on problems and methods in generalized assembly line balancing". *European Journal of Operational Research*, 168(3), 694-715, 2006.

[6] Battaïa O, Dolgui A. "A taxonomy of line balancing problems and their solution approaches". *International Journal of Production Economics*, 142(2), 259-277, 2013.

[7] Sivasankaran P, Shahabudeen P. "Literature review of assembly line balancing problems". *The International Journal of Advanced Manufacturing Technology*, 73(9-12), 1665–1694, 2014.

[8] Scholl A, Becker C. "State-of-the-art exact and heuristic solution procedures for simple assembly line balancing". *European Journal of Operational Research*, 168(3), 666–693, 2006.

[9] Scholl A. *Balancing and Sequencing of Assembly Lines*. Heidelberg, Physica-Verlag, 1999.

[10] Hackman ST, Magazine MJ, Wee TS. "Fast, effective algorithms for simple assembly line balancing problems". *Operations Research*, 37(6), 916-924, 1989.

[11] Klein R, Scholl A. "Maximizing the production rate in simple assembly line balancing-a branch and bound procedure". *European Journal of Operational Research*, 91(2), 367-385, 1996.

[12] Pastor R, Ferrer L. "An improved mathematical program to solve the simple assembly line balancing problem". *International Journal of Production Research*, 47(11), 2943–2959, 2009.

[13] Pastor R, Ferrer L, García A. *Evaluating optimization models to solve SALBP*. Editors: Gervasi O, Gavrilova ML. Computational Science and Its Applications–ICCSA 2007, 791-803, Berlin Heidelberg, Springer, 2007.

[14] Uğurdağ HF, Rachamadugu R, Papachristou CA. "Designing paced assembly lines with fixed number of stations". *European Journal of Operational Research*, 102(3), 488-501, 1997.

[15] Kilincci O. "A Petri net-based heuristic for simple assembly line balancing problem of type 2". *The International Journal of Advanced Manufacturing Technology*, 46(1), 329-338, 2010.

[16] Liu SB, Ong HL, Huang HC. "Two bi-directional heuristics for the assembly line type II problem". *The International Journal of Advanced Manufacturing Technology,* 22(9-10), 656–661, 2003.

[17] Anderson EJ, Ferris MC. "Genetic algorithms for combinatorial optimization: the assemble line balancing problem". *ORSA Journal on Computing*, 6(2), 161-173, 1994.

[18] Watanabe T, Hashimoto Y, Nishikawa I, Tokumaru H. "Line balancing using a genetic evolution model". *Control Engineering Practice*, 3(1), 69-76, 1995.

[19] Kim YJ, Kim YK, Cho Y. "A heuristic-based genetic algorithm for workload smoothing in assembly lines". *Computers & Operations Research*, 25(2), 99-111, 1998.

[20] Scholl A, Voß S. "Simple assembly line balancing-heuristic approaches". *Journal of Heuristics*, 2(3), 217-244, 1997.

[21] Chiang WC. "The application of a tabu search metaheuristic to the assembly line balancing problem". *Annals of Operations Research*, 77, 209-227, 1998.

[22] Henrici AA. *Comparison between simulated annealing and tabu search with an example from the production planning*. Editors: Dyckhoff H. et al. Operations Research Proceedings, 498–503, Springer Verlag, Berlin, Germany, 1994.

[23] Seyed-Alagheband SA, Fatemi Ghomi SMT, Zandieh M. "A simulated annealing algorithm for balancing the assembly line type II problem with sequence-dependent setup times between tasks". *International Journal of Production Research*, 49(3), 805-825, 2011.

[24] Boysen N, Fliedner M. "A versatile algorithm for assembly line balancing". *European Journal of Operational Research*, 184(1), 39-56, 2008.

[25] Zheng Q, Li M, Li Y, Tang Q. "Station ant colony optimization for the type 2 assembly line balancing problem". *The International Journal of Advanced Manufacturing Technology*, 66(9-12), 1859-1870, 2013.

[26] Nearchou AC. "Maximizing production rate and workload smoothing in assembly lines using particle swarm optimization". *International Journal of Production Economics*, 129(2), 242-250, 2011.

[27] Nearchou AC. "Balancing large assembly lines by a new heuristic based on differential evolution method". *The International Journal of Advanced Manufacturing Technology*, 34(9-10), 1016–1029, 2007.

[28] Zhang H, Yan Q, Liu Y, Jiang Z. "An integer-coded differential evolution algorithm for simple assembly line balancing problem of type 2". *Assembly Automation*, 36(3), 1-27, 2016.

[29] Blum C, Miralles C. "On solving the assembly line worker assignment and balancing problem via beam search". *Computers & Operations Research*, 38(1), 328-339, 2011.

[30] Blum C. "Iterative beam search for simple assembly line balancing with a fixed number of work stations". *SORT*, 35(2), 145-164, 2011.

[31] Lei D, Guo, X. "Variable neighborhood search for the second type of two-sided assembly line balancing problem". *Computers & Operations Research*, 72, 183-188.

[32] Polat O, Kalayci CB, Mutlu Ö, Gupta, SM. "A two-phase variable neighbourhood search algorithm for assembly line worker assignment and balancing problem type-II: an industrial case study". *International Journal of Production Research*, 54(3), 722-741, 2016.

[33] Dang QV, Pham K. "Design of a Footwear Assembly Line Using Simulation-based ALNS". Procedia CIRP, 40, 597-602, 2016.

[34] Shaw P. *Using constraint programming and local search methods to solve vehicle routing problems*. Editors: Maher M, Puget JF. Principles and Practice of Constraint Programming-CP98, 417–431, Berlin Heidelberg, Springer, 1998.

[35] Ropke S, Pisinger D. "An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows". *Transportation Science*, 40(4), 455-472, 2006.

[36] Pisinger D, Ropke S. *Large neighborhood search*. Editors: Gendreau M, Potvin JY. Handbook of Metaheuristics 2nd ed. New York, USA, Springer, 2010.

[37] Laporte G, Musmanno R, Vocaturo F. "An adaptive large neighbourhood search heuristic for the capacitated arc-routing problem with stochastic demands". *Transportation Science*, 44(1), 125-135, 2010.

[38] Muller LF, Spoorendonk S, Pisinger D. "A hybrid adaptive large neighborhood search heuristic for lot-sizing with setup times". *European Journal of Operational Research*, 218(3), 614-623, 2012.

[39] Demir E, Bektaş T, Laporte G. "An adaptive large neighborhood search heuristic for the pollution-routing problem". *European Journal of Operational Research*, 223(2), 346-359, 2012.

[40] Masson R, Lehuédé F, Péton O. "An adaptive large neighborhood search for the pickup and delivery problem with transfers". *Transportation Science*, 47(3), 344-355, 2013.

[41] Adulyasak Y, Cordeau JF, Jans R. "Optimization-based adaptive large neighborhood search for the production routing problem". *Transportation Science*, 48(1), 20-45, 2012.

[42] Belo-Filho MAF, Amorim P, Almada-Lobo B. "An adaptive large neighbourhood search for the operational integrated production and distribution problem of perishable products". *International Journal of Production Research*, 53(20), 6040-6058, 2015.

[43] Luo Z, Qin H, Zhang D, Lim A. "Adaptive large neighborhood search heuristics for the vehicle routing problem with stochastic demands and weight–related cost". *Transportation Research Part E: Logistics and Transportation Review*, 85, 69-89, 2016.

[44] Rifai AP, Nguyen HT, Dawal SZM. "Multi-objective adaptive large neighborhood search for distributed reentrant permutation flow shop scheduling". *Applied Soft Computing*, 40, 42-57, 2016.

[45] Leu YY, Matheson LA, Rees LP. "Assembly line balancing using genetic algorithms with heuristic generated initial populations and multiple criteria". *Decision Sciences*, 15, 581-606, 1994.

[46] Akpınar S, Bayhan GM. "A hybrid genetic algorithm for mixed model assembly line balancing problem with parallel workstations and zoning constraints". *Engineering Applications of Artificial Intelligence*, 24(3), 449-57, 2011.

[47] Helgeson NB, Birnie DP. "Assembly line balancing using the ranked positional weight technique". *Journal of Industrial Engineering*, 12(6), 394-398, 1961.

[48] Kim YK, Kim YJ, Kim Y. "Genetic algorithms for assembly line balancing with various objectives". *Computers & Industrial Engineering*, 30(3), 397–409, 1996.

[49] Goncalves JF, De Almeida JR. "A hybrid genetic algorithm for assembly line balancing". *Journal of Heuristics*, 8(6), 629-642, 2002.

[50] Akpınar, S. "Hybrid large neighbourhood search algorithm for capacitated vehicle routing problem". *Expert Systems with Applications*, 61, 28-38, 2016.