Path relinking for single row facility layout

Ravi Kothari
Diptesh Ghosh

W.P. No. 2012-05-01
May 2012

INDIAN INSTITUTE OF MANAGEMENT
AHMEDABAD – 380015
INDIA

Path relinking for single row facility layout

Ravi Kothari
Diptesh Ghosh

Abstract

The single row facility layout problem is the problem of arranging facilities with given lengths on a line, while minimizing the weighted sum of the distances between all pairs of facilities. The problem is NP-hard. In this paper we present path relinking algorithms to solve large sized instances of the problem. We use three different metaheuristics to generate sets of good quality initial solutions and subject the solutions in these sets to path relinking. We present our computational experience on 43 benchmark instances with up to 110 facilities.
Keywords: Facilities planning and design; Single Row Facility Layout, Tabu Search, Lin-Kernighan neighborhood search, Scatter search, Path relinking

1 Introduction

The single row facility layout problem (SRFLP) is a combinatorial optimization problem that has a large number of practical applications. It has been applied to model the layout of hotel corridors, departments in offices, warehouses and supermarkets, machines in flexible manufacturing systems, and assignment of files in computer disks. The SRFLP is described as follows. We are given a set $F = \{1, 2, \ldots, n\}$ of facilities, where facility j has length l_j, and each pair of facilities (i, j); $i, j \in F$ has a transmission intensity. The transmission intensity can be visualzed as the number of times the facilities in the pair need to communicate with each other. The cost of transmission between a pair of facilities (i, j) is defined as the product of the transmission intensity for the pair and the distance between the centroids of the two facilities in the pair. The objective of the problem is to arrange the facilities in a single row such that the sum of the costs of transmission among all pairs of facilities is minimized. Therefore a solution to the problem is a permutation of the facilities in F. The problem was first proposed in Simmons (1969) and is known to be NP-hard (Beghin-Picavet and Hansen 1982). It is a special case of the general space allocation problem (see Simmons 1969), and admits the minimum linear arrangement problem (see Petit 2003, Díaz et al. 2002) and the linear ordering problem (see Martí and Reinelt 2011) as special cases.

Since the SRFLP is NP-hard, solution approaches to the problem include both exact and heuristic methods. Exact methods include combinatorial branch and bound (Simmons 1969), mathematical programming based approaches (Love and Wong 1976, Heragu and Kusiak 1988, Amaral 2006; 2008), dynamic programming (Picard and Queyranne 1981, Kouvelis and Chiang 1992), cutting plane algorithms (Amaral 2009), semidefinite programming (Anjos et al. 2005, Anjos and Vannelli 2008, Anjos and Yen 2009, Hungerländer and Rendl 2011), and branch and cut (Amaral and Letchford 2012). However these methods have not yet been able to solve SRFLP instances with more than 42 facilities to optimality. Heuristics have been used for larger sized SRFLP instances.

The earliest heuristic approaches to solve the SRFLP resulted in the creation of construction heuristics which build up a solution by determining the location of one facility at a time (Heragu and Kusiak 1988, Kumar et al. 1995, Braglia 1997, Djellab and Gourgand 2001). However these heuristics do not perform well for large instances and have been superseded by improvement heuristics. Improvement heuristics start with one or more initial solutions and iteratively improve the

solution(s) until a stopping criterion is reached. Studies implementing improvement heuristics deal with simulated annealing (Romero and Sánchez-Flores 1990, Kouvelis and Chiang 1992, Heragu and Kusiak 1991, Heragu and Alfa 1992), tabu search (Gomes de Alvarenga et al. 2000, Samarghandi and Eshghi 2010), ant colony optimization (Solimanpur et al. 2005), particle swarm optimization (Samarghandi et al. 2010), scatter search (Kumar et al. 2008), and genetic algorithm (Datta et al. 2011). Among these, the results presented in (Datta et al. 2011) supersede all others for large sized SRFLP instances with up to 80 facilities.

Interestingly, the technique of path relinking which has proved effective on other combinatorial optimization problems (see, e.g., Glover et al. 2000) has not been applied to the SRFLP. In the current work therefore we form hybrid algorithms of path relinking with local search, tabu search, and scatter search and test the performance of these algorithms on large SRFLP instances.

The remainder of the paper is organized as follows. In the next section we present a brief introduction to the path relinking method. We follow this up in Section 3 with a description of several hybrid algorithms that we propose, all of which apply the path relinking principle to solutions obtained by other metaheuristics. We present results of our computational experiments with these algorithms in Section 4 and compare the results with the published literature. We summarize our contributions in Section 5 and point out the limitations in the current work.

2 Path Relinking

Path relinking (Glover 1977) is a search technique which starts with a set of good solutions to a problem and tries to explore the regions of the search space which have not been explored by the algorithm that generated the initial solutions. Algorithms to generate initial good quality solutions (like Tabu search, scatter search, GRASP etc.) coupled with path relinking strategies have proved to be very effective for a diverse set of combinatorial optimization problems.

To understand the concept of path relinking, let us assume that we have a population P of incumbent solutions, and an algorithm $\mathcal{A}$ that generated the incumbent solutions. Path relinking is a generalized path construction technique which uses strategic design and the attribute information of solutions in P to generate additional solutions which may not have been considered by the algorithm $\mathcal{A}$, and which may have better objective function values than the solutions in P. The path relinking technique chooses two solutions from P, marks one as the initiating solution and the other as the guiding solution. It then generates a path from the initiating solution to the guiding solution through a set of moves. A move starts with a solution and generates another solution in which some attribute of the initiating solution is replaced with some attribute of the guiding solution. The idea is to introduce attributes of the guiding solution in each move to finally reach the guiding solution from the initiating solution or to extend beyond it (extrapolated relinking, see Glover et al. 2000). The new solutions that are generated share a significant subset of attributes of both the initiating solution and the guiding solution in varying mixes depending on the path selected. Pictorially, this is shown in Figure 1, where the dots correspond to solutions, Π^* is the initiating solution and Π^{**} is the guiding solution. The solid line shows the trajectory of the algorithm ($\mathcal{A}$) when the initiating and guiding solutions are generated, and the dotted line shows the path relinked solutions. The roles of initiating solution and the guiding solutions are interchangeable and interchanging them may lead to different paths in the search space.

A path relinking algorithm is iterative and proceeds as follows. It starts with a set of solutions. If all the solutions in the set are identical at the beginning of an iteration, then there is no path to be relinked and the algorithm outputs a solution in the set and terminates. Otherwise it chooses two non-identical solutions from the set, marks one as the initiating solution and the other as a guiding solution. It then generates a path from the initiating solution to the guiding solution (and beyond, if extrapolated relinking is used), and chooses the best solution encountered in the path. If this

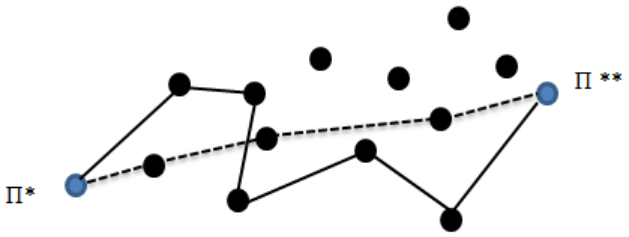

Figure 1: Path relinking between two permutations on same search path

solution is better than the worst solution in the set, then the worst solution in the set is replaced with it and the iteration is complete.

3 Our path relinking algorithms for the SRFLP

We present four algorithms which implement path relinking for the SRFLP. The path relinking mechanism in all the four algorithms is identical, and the algorithms vary only in terms of the heuristic used to generate the initial solutions for path relinking. In this section, we first present the path relinking mechanism for the SRFLP and then follow it up with a brief description of the heuristics used to generate the set of initial solution.

3.1 Relinking paths

Solutions in a single row facility layout problem are permutations of facilities. A solution is said to be better than another if the cost of the former is lower than that of the latter. In our path relinking algorithms, the initial solutions generated by a heuristic are sorted in non-decreasing order of their costs. Every solution in the first half of the sorted list is used once as an initiating solution. For an initiating solution the guiding solution is that solution in the second half of the list which has the maximum deviation distance (see Sörensen 2007) from the initiating solution. A path between the initiating solution and the guiding solution is obtained by executing moves that are necessary to transform the initiating solution into guiding solution. For example, if in an intermediate solution in the path, a facility π_i is in the i-th position in the permutation, and in the guiding solution π_i is in the j-th position in the permutation, then a move will interchange π_i with the facility in the j-th position in the permutation corresponding to the intermediate solution. We illustrate this method of path relinking with the following example.

Consider a SRFLP with seven facilities where $\Pi^* = \{4, 5, 3, 1, 2, 7, 6\}$ is the intiating solution and $\Pi^{**} = \{2, 5, 1, 3, 4, 6, 7\}$ is the guiding solution. The relinking is done as follows. The first facility in Π^{**} is 2 and it is located at fifth position in Π^*. So in the first move, we interchange the facilities in the first and fifth positions in Π^* to generate the first intermediate permutation $\Pi(1) = \{2, 5, 3, 1, 4, 6, 7\}$. We next consider the facility in the second position in Π^{**}, i.e., 5. This facility is located at the same position in $\Pi(1)$ and so we do not generate a move to reposition this facility. The facility in the third position in Π^{**} is 1 and it is located at fourth position in $\Pi(1)$. So we interchange the third and the fourth facilities in $\Pi(1)$ to obtain the next intermediate permutation $\Pi(2) = \{2, 5, 1, 3, 4, 6, 7\}$. Continuing this process all the intermediate permutations are obtained till the algorithm obtains Π^{**}. All the intermediate permutations are listed below in Table 1.

Table 1: Intermediate permutations in the path from $\Pi^* = \{4, 5, 3, 1, 2, 7, 6\}$ to $\Pi^{**} = \{2, 5, 1, 3, 4, 6, 7\}$

No.	Name	Type	Permutation
1.	Π^*	Initiating	$\{4, 5, 3, 1, 2, 7, 6\}$
2.	$\Pi(1)$	Intermediate	$\{2, 5, 3, 1, 4, 7, 6\}$
3.	$\Pi(2)$	Intermediate	$\{2, 5, 1, 3, 4, 7, 6\}$
4.	Π^{**}	Guiding	$\{2, 5, 1, 3, 4, 6, 7\}$

Thus, in our algorithms we do not implement extrapolated relinking. This is because we do not see any method of naturally extending the path that we construct between the initiating and guiding solutions.

The last step in the relinking process is to choose a solution in the relinked path that can possibly replace a solution in the set of solutions. In our algorithms this is done as follows. As a first step, we choose every second solution along the relinked path and subject it to local search using the insertion neighborhood (see Heragu and Alfa 1992, for an introduction to insertion neighborhoods for the SRFLP). If the locally optimal solution thus obtained is better than the solution in the path we replace the solution in the path with the locally optimal solution. This results in a modified set of solutions "along the relinked path". We then look at the modified set of solutions and choose the best solution in the set as the output of the algorithm.

3.2 Generating the set of initial solutions

As mentioned earlier, our algorithms differ from each other in terms of the method used to generate initial solutions. Our first two algorithms, which we call PR-TS2OPT and PR-TSINSERT generate initial solutions using tabu search with a 2-opt neighborhood and insertion neighborhood respectively. Our third algorithm, which we call PR-LKINSERT generates initial solutions using local search with an insertion neighborhood structure. Other than these there are four path relinking algorithms, PR-SS1A, PR-SS1P, PR-SS2A, and PR-SS2P which use scatter search variants SS1A, SS1P, SS2A, and SS2P reported in Kothari and Ghosh (2012b) to generate the set of initial solutions. We now briefly describe these methods of initial solution generation.

Tabu search with 2-opt neighborhood A 2-opt neighbor of a solution is one obtained by interchanging the locations of exactly two facilities in the solution. A 2-opt neighborhood of a solution is the set of all its 2-opt neighbors. Tabu search using the 2-opt neighborhood has been reported in Samarghandi et al. (2010) and Kothari and Ghosh (2012c). Both these studies implement tabu search in a multi-start manner. Thus both tabu search algorithms terminate with a set of solutions which can form the set of initial solutions for path relinking. In PR-TS2OPT we use the implementation in Kothari and Ghosh (2012c) to generate the set of initial solutions for path relinking.

Tabu search with insertion neighborhood An insertion neighbor of a solution is obtained by removing one facility from the solution and re-introducing it at some other location in the solution. An insertion neighborhood of a solution is the set of all its insertion neighbors. Tabu search using the insertion neighborhood has been reported in Kothari and Ghosh (2012c). This tabu search is also implemented in a multi-start manner and terminates with a set of solutions that can be used for path relinking. In PR-TSINSERT we use the tabu search implementation in Kothari and Ghosh (2012c) to create the set of initial solutions for path relinking.

Lin-Kernighan local search with insertion neighborhood The Lin-Kernighan neighborhood (see, e.g., Lin and Kernighan 1973) is a variable depth neighborhood in which moves are composite in nature. A composite move in the neighborhood comprises of components which are basic moves like 2-opt and insertion moves. The number of basic moves that make up a composite move depends on the improvement in cost obtained by performing the move. A local search algorithm based on Lin-Kernighan moves called LKINSERT has been proposed in Kothari and Ghosh (2012a). It is a multi-start local search algorithm which at termination yields a set of solutions which can be used as starting solutions for the path relinking algorithm called PR-LKINSERT.

Scatter search Scatter search (see, e.g., Glover et al. 2000) is a population heuristic in which a population of solutions are iteratively improved to generate good quality solutions. In each iteration of scatter search, subsets of solutions are generated, and the solutions in these subsets are combined to generate new solutions. These solutions are then examined to see whether their costs allow them to be introduced in the population of solutions that the next iteration of scatter search will consider. Four scatter search algorithms called SS1A, SS1P, SS2A, and SS2P have been presented in Kothari and Ghosh (2012b). We use these implementations to generate initial solutions for path relinking algorithms called PR-SS1A, PR-SS1P, PR-SS2A, and PR-SS2P respectively. An important difference between these algorithms and the other three algorithms that we propose (PR-TS2OPT, PR-TSINSERT, and PR-LKINSERT) is that in these, path relinking is carried out after each scatter search iteration.

In the next section we present results of experiments we performed on the path relinking algorithms on benchmark SRFLP instances.

4 Computational experience

We coded the algorithms presented in Section 3 in C and ran them on an Intel machine with 4GB RAM and four i5-2500 processors at 3.30GHz running Windows 7. We used the gcc 4.6.1 compiler to compile our programs. The size of the set of initial solutions were set to $\lfloor 2n/3 \rfloor$ for PR-TS2OPT, PR-TSINSERT, and PR-LKINSERT where n is the size of the problem instance, and it was 20 for the other four path relinking algorithms. We used 43 benchmark instances to compare the performance of our algorithms with the best results known in the published literature.

The first set of 20 instances were first reported in Anjos et al. (2005) and consists of five instances each of sizes 60, 70, 75, and 80. Computational results for these instances are available in the published literature (see, e.g., Anjos et al. 2005, Samarghandi and Eshghi 2010, Datta et al. 2011). Among these, the results in Datta et al. (2011) supersede those in all other studies. We present a computational comparison of our path relinking algorithms on the Anjos instances in Tables 2 and 3. In Table 2 we present the results obtained in Datta et al. (2011) and those obtained by the hybrids of path relinking and tabu search and local search. The first two columns of the table identify the instances. The third column presents the results reported in Datta et al. (2011) under the label DA&F. The next six columns report the results obtained by TS2OPT, PR-TS2OPT, TSINSERT, PR-TSINSERT, LKINSERT, and PR-LKINSERT respectively. In Table 3 we present the results obtained in Datta et al. (2011) and those obtained by the hybrids of path linking and various scatter search algorithms. The first three columns are identical to those in Table 2. The next eight columns report the results obtained by SS1A, PR-SS1A, SS1P, PR-SS1P, SS2A, PR-SS2A, SS2P, and PR-SS2P respectively.

From the tables we observe that all the path relinking algorithms produced identical solutions for all the 20 instances in the set. In six of the instances the solutions by our path relinking algorithms were better than those in the published literature. However these solutions had costs which were

Table 2: Performance of path relinking algorithms using tabu search and local search on the instances in Anjos et al. (2005)

Instance	Size	DA&F	TS2OPT	PR-TS2OPT	TSINSERT	PR-TSIINSERT	LKINSERT	PR-LKIINSERT
Anjos-60-01	60	1477834.0	1477834.0	1477834.0	1477834.0	1477834.0	1477834.0	1477834.0
Anjos-60-02	60	841776.0	841790.0	841776.0	841776.0	841776.0	841776.0	841776.0
Anjos-60-03	60	648337.5	648337.5	648337.5	648337.5	648337.5	648337.5	648337.5
Anjos-60-04	60	398406.0	398406.0	398406.0	398406.0	398406.0	398406.0	398406.0
Anjos-60-05	60	318805.0	318805.0	318805.0	318805.0	318805.0	318805.0	318805.0
Anjos-70-01	70	1528560.0	1528604.0	1528537.0	1528537.0	1528537.0	1528537.0	1528537.0
Anjos-70-02	70	1441028.0	1440128.0	1441028.0	1441028.0	1441028.0	1441028.0	1441028.0
Anjos-70-03	70	1518993.5	1518993.0	1518993.5	1518993.5	1518993.5	1518993.5	1518993.5
Anjos-70-04	70	968796.0	968796.0	968796.0	968796.0	968796.0	968796.0	968796.0
Anjos-70-05	70	4218017.5	4218002.0	4218002.5	4218002.5	4218002.5	4218002.5	4218002.5
Anjos-75-01	75	2393456.5	2393456.5	2393456.5	2393456.5	2393456.5	2393456.5	2393456.5
Anjos-75-02	75	4321190.0	4321190.0	4321190.0	4321190.0	4321190.0	4321190.0	4321190.0
Anjos-75-03	75	1248537.0	1248607.0	1248423.0	1248423.0	1248423.0	1248423.0	1248423.0
Anjos-75-04	75	3941845.5	3941816.5	3941816.5	3941816.5	3941816.5	3941816.5	3941816.5
Anjos-75-05	75	1791408.0	1791408.0	1791408.0	1791408.0	1791408.0	1791408.0	1791408.0
Anjos-80-01	80	2069097.5	2069145.5	2069097.5	2069097.5	2069097.5	2069097.5	2069097.5
Anjos-80-02	80	1921177.0	1921136.0	1921136.0	1921136.0	1921136.0	1921136.0	1921136.0
Anjos-80-03	80	3251368.0	3251368.0	3251368.0	3251368.0	3251368.0	3251368.0	3251368.0
Anjos-80-04	80	3746515.0	3746515.0	3746515.0	3746515.0	3746515.0	3746515.0	3746515.0
Anjos-80-05	80	1588901.0	1588885.0	1588885.0	1588885.0	1588885.0	1588885.0	1588885.0

Table 3: Performance of path relinking algorithms using scatter search on the instances in Anjos et al. (2005)

Instance	Size	DA&F	SS1A	PR-SS1A	SS1P	PR-SS1P	SS2A	PR-SS2A	SS2P	PR-SS2P
Anjos-60-01	60	1477834.0	1477834.0	1477834.0	1477834.0	1477834.0	1477834.0	1477834.0	1477834.0	1477834.0
Anjos-60-02	60	841776.0	841776.0	841776.0	841776.0	841776.0	841776.0	841776.0	841776.0	841776.0
Anjos-60-03	60	648337.5	648337.5	648337.5	648337.5	648337.5	648337.5	648337.5	648337.5	648337.5
Anjos-60-04	60	398406.0	398406.0	398406.0	398406.0	398406.0	398406.0	398406.0	398406.0	398406.0
Anjos-60-05	60	318805.0	318805.0	318805.0	318805.0	318805.0	318805.0	318805.0	318805.0	318805.0
Anjos-70-01	70	1528560.0	1528537.0	1528537.0	1528537.0	1528537.0	1528537.0	1528537.0	1528537.0	1528537.0
Anjos-70-02	70	1441028.0	1441028.0	1441028.0	1441028.0	1441028.0	1441028.0	1441028.0	1441028.0	1441028.0
Anjos-70-03	70	1518993.5	1518993.5	1518993.5	1518993.5	1518993.5	1518993.5	1518993.5	1518993.5	1518993.5
Anjos-70-04	70	968796.0	968796.0	968796.0	968796.0	968796.0	968796.0	968796.0	968796.0	968796.0
Anjos-70-05	70	4218017.5	4218002.5	4218002.5	4218002.5	4218002.5	4218002.5	4218002.5	4218002.5	4218002.5
Anjos-75-01	75	2393456.5	2393456.5	2393456.5	2393456.5	2393456.5	2393456.5	2393456.5	2393456.5	2393456.5
Anjos-75-02	75	4321190.0	4321190.0	4321190.0	4321190.0	4321190.0	4321190.0	4321190.0	4321190.0	4321190.0
Anjos-75-03	75	1248537.0	1248423.0	1248423.0	1248423.0	1248423.0	1248423.0	1248423.0	1248423.0	1248423.0
Anjos-75-04	75	3941845.5	3941816.5	3941816.5	3941816.5	3941816.5	3941816.5	3941816.5	3941816.5	3941816.5
Anjos-75-05	75	1791408.0	1791408.0	1791408.0	1791408.0	1791408.0	1791408.0	1791408.0	1791408.0	1791408.0
Anjos-80-01	80	2069097.5	2069097.5	2069097.5	2069097.5	2069097.5	2069097.5	2069097.5	2069097.5	2069097.5
Anjos-80-02	80	1921177.0	1921136.0	1921136.0	1921136.0	1921136.0	1921136.0	1921136.0	1921136.0	1921136.0
Anjos-80-03	80	3251368.0	3251368.0	3251368.0	3251368.0	3251368.0	3251368.0	3251368.0	3251368.0	3251368.0
Anjos-80-04	80	3746515.0	3746515.0	3746515.0	3746515.0	3746515.0	3746515.0	3746515.0	3746515.0	3746515.0
Anjos-80-05	80	1588901.0	1588885.0	1588885.0	1588885.0	1588885.0	1588885.0	1588885.0	1588885.0	1588885.0

already obtained by other algorithms (e.g., the scatter search algorithms in Kothari and Ghosh (2012b)). Path relinking was able to improve the solutions generated by TS2OPT in four of the 20 instances. However, the path relinking algorithms could not improve the best solutions output by TSINSERT, LKINSERT, and all the scatter search variants on any of 20 instances.

The second set of benchmark instances also consists of 20 instances. These were first reported in Anjos and Yen (2009) and consist of five instances each of sizes 64, 72, 81, and 100. The connection data in these instances are from the sko benchmark instances for the quadratic assignment problem and hence these instances are known as the sko instances. In the published literature, results from computations on these instances have been reported in Anjos and Yen (2009), Amaral and Letchford (2012). Among these, the results in Amaral and Letchford (2012) supersede the other. In Tablea 4 and 5 we present a comparison of the results obtained in Amaral and Letchford (2012) under the label A&L and those obtained by the path relinking algorithms presented in this paper. The structure of Table 4 is similar to that of Table 2 and the structure of Table 5 is similar to that of Table 3.

The results in these two tables show that most of the path relinking algorithms obtained solutions that were better than those reported in the literature for 18 of the 20 instances. In the other two instances the costs of the solutions from the path relinking algorithms matched the costs of the best solutions known in the literature. The hybrid algorithm PR-TS2OPT improved the solutions output by TS2OPT in 16 instances, PR-TSINSERT improved the solutions output by TSINSERT in 15 instances, PR-LKINSERT improved the solutions output by LKINSERT in 11 instances, and PR-SS1A improved the solutions output by SS1A in two instances. The solutions output by the other three hybrid path relinking algorithms matched the solutions output by the original scatter search heuristics in all the 20 instances. However, the path relinked heuristics were not able to output solutions that were superior to the best ones from the scatter search algorithms on any of the instances.

The last set of benchmark instances consists of three instances of size 110. These were proposed in Letchford and Amaral (2011), where upper bounds to the costs of optimal solutions were reported. In Tables 6 and 7 we compare the results obtained by the path relinking algorithms on these problems to those reported in Letchford and Amaral (2011). The structure of Table 6 is identical to that of Table 4 and the structure of Table 7 is similar to that of Table 5.

We see from these tables that the path relinking algorithms output solutions whose costs are lower than those reported in Letchford and Amaral (2011) on all the three instances. The hybrid path relinking algorithms using tabu search with 2-opt and insertion neighborhoods, and local search using the Lin-Kernighan neighborhoods output better solutions than those output by the corresponding tabu search and local search algorithms. However, they were not able to improve any solution output by any version of scatter search for these instances.

Based on all the computational experiments, we conclude that path relinking is a possible method for improving solutions obtained using tabu search and local search algorithms. However they are not very effective in improving solutions obtained using scatter search algorithms.

5 Summary of the work

In this paper, we examined the possibility of using path relinking combined with other metaheuristics to obtain good quality solutions to large instances of the single row facility layout problem. In particular, we used two multi-start tabu search algorithms, TS2OPT and TSINSERT (see Kothari and Ghosh 2012c), one multi-start local search algorithm LKINSERT using a Lin-Kernighan neighborhood (see Kothari and Ghosh 2012a), and four variants of scatter search SS1A, SS1P, SS2A, and SS2P (see Kothari and Ghosh 2012b). We compared the solutions output by the hybrid path relinking algorithms with the best available in the published literature.

Table 4: Performance of path relinking algorithms using tabu search and local search on the sko instances in Anjos and Yen (2009)

Instance	Size	A&L	TS2OPT	PR-TS2OPT	TSINSERT	PR-TSIINSERT	LKINSERT	PR-LKIINSERT
sko-64-01	64	96930.0	96915.0	96919.0	96969.0	96891.0	96933.0	96886.0
sko-64-02	64	634332.5	634563.5	634332.5	634595.5	634332.5	634338.5	634332.5
sko-64-03	64	414356.5	414327.5	414323.5	414338.5	414323.5	414323.5	414323.5
sko-64-04	64	297358.0	297332.0	297137.0	297399.0	297137.0	297205.0	297137.0
sko-64-05	64	501922.5	501922.5	501922.5	501922.5	501922.5	501922.5	501922.5
sko-72-01	72	139174.0	139195.0	131950.0	139179.0	139161.0	139150.0	139150.0
sko-72-02	72	712261.0	712011.0	711998.0	712217.0	711998.0	712005.0	712005.0
sko-72-03	72	1054184.5	1054110.5	1054110.5	1054110.5	1054110.5	1054110.5	1054110.5
sko-72-04	72	920693.5	920086.5	919615.5	921268.5	919615.5	919635.5	919586.5
sko-72-05	72	428305.5	428617.5	428228.5	428248.5	428244.5	428879.5	428228.5
sko-81-01	81	205475.0	205161.0	205123.0	205145.0	205150.0	205166.0	205180.0
sko-81-02	81	523021.5	521399.5	521391.5	521402.5	521391.5	521391.5	521391.5
sko-81-03	81	970920.0	971169.0	970796.0	970912.0	970796.0	970862.0	970827.0
sko-81-04	81	2032634.0	2032361.0	2031833.0	2032143.0	2031919.0	2031979.0	2031993.0
sko-81-05	81	1303756.0	1304266.0	1302993.0	1302833.0	1302711.0	1303805.0	1302711.0
sko-100-01	100	378584.0	378626.0	378268.0	378634.0	378307.0	378614.0	378336.0
sko-100-02	100	2076714.5	2076231.5	2076034.5	2076023.5	2076048.5	2076048.5	2076096.5
sko-100-03	100	16177226.5	16159760.0	16145896.0	16149000.0	16149444.0	16148818.0	16145598.0
sko-100-04	100	3237111.0	3238812.0	3232565.0	3233362.0	3232749.0	3232740.0	3232522.0
sko-100-05	100	1034922.5	1033338.5	1033341.5	1033421.5	1033346.0	1033345.5	1033291.5

Table 5: Performance of path relinking algorithms using scatter search on the sko instances in Anjos and Yen (2009)

Instance	Size	A&L	SS1A	PR-SS1A	SS1P	PR-SS1P	SS2A	PR-SS2A	SS2P	PR-SS2P
sko-64-01	64	96930.0	96884.0	96884.0	96883.0	96883.0	96884.0	96884.0	96890.0	96890.0
sko-64-02	64	634332.5	634338.5	634338.5	634332.5	634332.5	634332.5	634332.5	634332.5	634332.5
sko-64-03	64	414356.5	414323.5	414323.5	414323.5	414323.5	414323.5	414323.5	414323.5	414323.5
sko-64-04	64	297358.0	297129.0	297129.0	297129.0	297129.0	297137.0	297137.0	297129.0	297129.0
sko-64-05	64	501922.5	501922.5	501922.5	501922.5	501922.5	501922.5	501922.5	501922.5	501922.5
sko-72-01	72	139174.0	139153.0	139153.0	139150.0	139150.0	139150.0	139150.0	139150.0	139150.0
sko-72-02	72	712261.0	711998.0	711998.0	711998.0	711998.0	711998.0	711998.0	711998.0	711998.0
sko-72-03	72	1054184.5	1054141.5	1054141.5	1054110.5	1054110.5	1054110.5	1054110.5	1054110.5	1054110.5
sko-72-04	72	920693.5	919586.5	919586.5	919586.5	919586.5	919586.5	919586.5	919586.5	919586.5
sko-72-05	72	428305.5	428226.5	428226.5	428226.5	428226.5	428228.5	428226.5	428228.5	428226.5
sko-81-01	81	205475.0	205112.0	205112.0	205106.0	205106.0	205120.0	205112.0	205112.0	205112.0
sko-81-02	81	523021.5	521391.5	521391.5	521391.5	521391.5	521391.5	521391.5	521391.5	521391.5
sko-81-03	81	970920.0	970796.0	970796.0	970796.0	970796.0	970796.0	970796.0	970796.0	970796.0
sko-81-04	81	2032634.0	2031803.0	2031803.0	2031803.0	2031803.0	2031803.0	2031803.0	2031803.0	2031803.0
sko-81-05	81	1303756.0	1302711.0	1302711.0	1302711.0	1302711.0	1302711.0	1302711.0	1302711.0	1302711.0
sko-100-01	100	378584.0	378249.0	378239.0	378234.0	378234.0	378259.0	378259.0	378234.0	378234.0
sko-100-02	100	2076714.5	2076008.5	2076008.5	2076008.5	2076008.5	2076008.5	2076008.5	2076008.5	2076008.5
sko-100-03	100	16177226.5	16145598.0	16145598.0	16145614.0	16145614.0	16145598.0	16145598.0	16149444.0	16149444.0
sko-100-04	100	3237111.0	3232522.0	3232522.0	3232531.0	3232531.0	3232522.0	3232522.0	3232522.0	3232522.0
sko-100-05	100	1034922.5	1033085.5	1033080.5	1033080.5	1033080.5	1033085.5	1033085.5	1033080.5	1033080.5

Table 6: Performance of path relinking algorithms using tabu search and local search on the instances in Letchford and Amaral (2011)

Instance	Size	L&A	TS2OPT	PR-TS2OPT	TSINSERT	PR-TSIINSERT	LKINSERT	PR-LKIINSERT
Amaral-110-01	110	144331884.5	144369632.0	144297632.0	144386928.0	144296768.0	144330992.0	144297712.0
Amaral-110-02	110	86065390.0	86073800.0	86050544.0	86071808.0	86050800.0	86060416.0	86050384.0
Amaral-110-03	110	2234803.5	2235559.5	2234811.5	2234871.5	2234800.5	2234825.5	2234781.5

Table 7: Performance of path relinking algorithms using scatter search on the instances in Letchford and Amaral (2011)

Instance	Size	L&A	SS1A	PR-SS1A	SS1P	PR-SS1P	SS2A	PR-SS2A	SS2P	PR-SS2P
Amaral-110-01	110	144331884.5	144296768.0	144296768.0	144297440.0	144297440.0	144296768.0	144296768.0	144296768.0	144296768.0
Amaral-110-02	110	86065390.0	86050112.0	86050112.0	86050208.0	86050208.0	86050112.0	86050112.0	86050112.0	86050112.0
Amaral-110-03	110	2234803.5	2234743.5	2234743.5	2234798.5	2234798.5	2234743.5	2234743.5	2234743.5	2234743.5

We performed experiments using 43 benchmark instances for the SRFLP with sizes varying between 60 and 110. These are the largest size benchmark instances available in the literature. Our experiments showed that the path relinking algorithms produced solutions that were better than those in the published literature for 25 of the 43 instances. Most of these instances were the larger sized ones among the benchmark instances. In the others, the solutions they produced had costs equal to those of the best known solutions from the literature. We observed that path relinking was effective in improving solutions obtained using tabu search and local search, but were not effective in general for improving solutions obtained using scatter search algorithms. The solutions that they output when they form hybrid algorithms with tabu search and local search are those that scatter search normally outputs without path relinking. Hence path relinking is a good strategy to hybridize with heuristic algorithms other than scatter search for the single row facility layout problem.

References

Amaral, A. R. S. (2006). On the exact solution of a facility layout problem. *European Journal of Operational Research*, 173(2):508–518.

Amaral, A. R. S. (2008). An Exact Approach to the One-Dimensional Facility Layout Problem. *Operations Research*, 56(4):1026–1033.

Amaral, A. R. S. (2009). A new lower bound for the single row facility layout problem. *Discrete Applied Mathematics*, 157(1):183–190.

Amaral, A. R. S. and Letchford, A. N. (2012). A polyhedral approach to the single row facility layout problem. *Mathematical Programming*. Available at http://dx.doi.org/10.1007/s10107-012-0533-z.

Anjos, M. F., Kennings, A., and Vannelli, A. (2005). A semidefinite optimization approach for the single-row layout problem with unequal dimensions. *Discrete Optimization*, 2(2):113–122.

Anjos, M. F. and Vannelli, A. (2008). Computing Globally Optimal Solutions for Single-Row Layout Problems Using Semidefinite Programming and Cutting Planes. *INFORMS Journal on Computing*, 20(4):611–617.

Anjos, M. F. and Yen, G. (2009). Provably near-optimal solutions for very large single-row facility layout problems. *Optimization Methods and Software*, 24(4-5):805–817.

Beghin-Picavet, M. and Hansen, P. (1982). Deux problèmes daffectation non linéaires. *RAIRO, Recherche Opérationnelle*, 16(3):263–276.

Braglia, M. (1997). Heuristics for single-row layout problems in flexible manufacturing systems. *Production Planning & Control*, 8(6):558–567.

Datta, D., Amaral, A. R. S., and Figueira, J. R. (2011). Single row facility layout problem using a permutation-based genetic algorithm. *European Journal of Operational Research*, 213(2):388–394.

Díaz, J., Petit, J., and Serna, M. (2002). A survey of graph layout problems. *ACM Computing Surveys*, 34(3):313–356.

Djellab, H. and Gourgand, M. (2001). A new heuristic procedure for the single -row facility layout problem. *International Journal of Computer Integrated Manufacturing*, 14(3):270–280.

Glover, F. (1977). Heuristics for integer programming using surrogate constraints. *Decision Sciences*, 8(1):156–166.

Glover, F., Laguna, M., and Martí, R. (2000). Fundamentals of scatter search and path relinking. *Control and Cybernetics*, 39:653–684.

Gomes de Alvarenga, A., Negreiros-Gomes, F. J., and Mestria, M. (2000). Metaheuristic methods for a class of the facility layout problem. *Journal of Intelligent Manufacturing*, 11:421–430.

Heragu, S. S. and Alfa, A. S. (1992). Experimental analysis of simulated annealing based algorithms for the layout problem. *European Journal of Operational Research*, 57(2):190–202.

Heragu, S. S. and Kusiak, A. (1988). Machine Layout Problem in Flexible Manufacturing Systems. *Operations Research*, 36(2):258–268.

Heragu, S. S. and Kusiak, A. (1991). Efficient models for the facility layout problem. *European Journal Of Operational Research*, 53:1–13.

Hungerländer, P. and Rendl, F. (Unpublished results, 2011). A computational study for the single-row facility layout problem. Available at www.optimization-online.org/DB_FILE/2011/05/3029.pdf.

Kothari, R. and Ghosh, D. (2012a). A Lin-Kernighan heuristic for single row facility layout (w.p. no. 2012-01-04). Ahmedabad, India: IIM Ahmedabad, Production & Quantitative Methods. Available at www.optimization-online.org/DB_HTML/2012/01/3315.html.

Kothari, R. and Ghosh, D. (2012b). Scatter search algorithms for the single row facility layout problem (w.p. no. 2012-04-01). Ahmedabad, India: IIM Ahmedabad, Production & Quantitative Methods. Available at www.optimization-online.org/DB_HTML/2012/03/3400.html.

Kothari, R. and Ghosh, D. (2012c). Tabu search for the single row facility layout problem using exhaustive 2-opt and insertion neighborhoods (w.p. no. 2012-01-03). Ahmedabad, India: IIM Ahmedabad, Production & Quantitative Methods. Available at www.optimization-online.org/DB_HTML/2012/01/3314.html.

Kouvelis, P. and Chiang, W.-C. (1992). A simulated annealing procedure for single row layout problems in flexible manufacturing systems. *International Journal of Production Research*, 30(4):717–732.

Kumar, R. K., Hadjinicola, G. C., and Lin, T.-L. (1995). A heuristic procedure for the single-row facility layout problem. *European Journal of Operational Research*, 87(1):65–73.

Kumar, S., Asokan, P., Kumanan, S., and Varma, B. (2008). Scatter search algorithm for single row layout problem in fms. *Advances in Production Engineering & Management*, 3(4):193–204.

Letchford, A. N. and Amaral, A. R. S. (2011). A polyhedral approach to the single row facility layout problem. Technical report, The Department of Management Science, Lancaster University.

Lin, S. and Kernighan, B. W. (1973). An Effective Heuristic Algorithm for the Traveling-Salesman Problem. *Operations Research*, 21(2):498–516.

Love, R. F. and Wong, J. Y. (1976). On solving a one-dimensional space allocation problem with integer programming. *INFOR*, 14(2):139–144.

Martí, R. and Reinelt, G. (2011). *The Linear Ordering Problem*. Springer-Verlag Berlin Heidelberg.

Petit, J. (2003). Experiments on the minimum linear arrangement problem. *J. Exp. Algorithmics*, 8.

Picard, J.-C. and Queyranne, M. (1981). On the one-dimensional space allocation problem. *Operations Research*, 29(2):371–391.

Romero, D. and Sánchez-Flores, A. (1990). Methods for the one-dimensional space allocation problem. *Computers & Operations Research*, 17(5):465–473.

Samarghandi, H. and Eshghi, K. (2010). An efficient tabu algorithm for the single row facility layout problem. *European Journal of Operational Research*, 205(1):98–105.

Samarghandi, H., Taabayan, P., and Jahantigh, F. F. (2010). A particle swarm optimization for the single row facility layout problem. *Computers & Industrial Engineering*, 58(4):529–534.

Simmons, D. M. (1969). One-Dimensional Space Allocation: An Ordering Algorithm. *Operations Research*, 17(5):812–826.

Solimanpur, M., Vrat, P., and Shankar, R. (2005). An ant algorithm for the single row layout problem in flexible manufacturing systems. *Computers & Operations Research*, 32(3):583–598.

Sörensen, K. (2007). Distance measures based on the edit distance for permutation-type representations. *Journal of Heuristics*, 13:35–47. 10.1007/s10732-006-9001-3.