ENHANCED LOCAL SEARCH APPLIED TO THE SINGLE-ROW FACILITY LAYOUT PROBLEM

André R. S. Amaral
Departamento de Informática, Universidade Federal do Espírito Santo (UFES),
Vitória, ES 29060-970, Brazil
amaral@inf.ufes.br

Abstract
In this paper, a method called an *enhanced local search* (ELS) is described. ELS is a general framework endowed with a number of local search algorithms intended for providing a better performance for a wide range of problem instances. Thus, ELS is expected to perform well regardless of the particular instances of the problem to be solved. Computational tests show that ELS can be very efficient when applied to single row facility layout problems.

Keywords: Facility Layout, Local search.

1. Introduction

Suppose that there is a number of *local search procedures* (also called *iterative improvement algorithms*) which can be applied to a given optimization problem. Let $\mathscr{H}$ be a set containing these local search procedures. Let $\mathscr{I}$ be a selected subset of $\mathscr{H}$. The subset $\mathscr{I}$ (sometimes called a *pool of local search procedures*) can be stored in a database, which is to be explored by an *enhanced local search* (ELS), i.e. a method that outputs the best solution in the set of solutions provided by the local search procedures in $\mathscr{I}$. An enhanced local search approach can be specialized so as to supervise the solution process, managing the several local search procedures in its database, and determining which actions may be taken in order to improve the solution process (e.g. aborting one local search procedure that has exceeded a given computing time limit).

ELS is based on the following philosophy: rather than selecting one local search with performance superior to all others, one could proceed to the construction of a more general framework endowed with a number of local search algorithms. For each local search, there might be an input for which the local search performs very badly (e.g. getting trapped in an undesirable local optimum, excessive time/memory requirements), but the instance, which causes a poor behavior in one local search procedure, may not cause that poor behavior in another. Thus, by diversifying across a number of different local search procedures, one benefits from the distinctive strengths of each. In this sense, ELS can be seen as a strategy that compensates individual local search performances for a wide range of problem instances. The set $\mathscr{I}$ has to be formed so as to reinforce this compensatory mechanism. A computational aspect of ELS is that it is less expensive to compute the solutions for a small subset $\mathscr{I}$ of $\mathscr{H}$. A theoretical aspect of the technique is whether it is possible to obtain a smallest subset $\mathscr{I}$ that will provide the same results as the whole set $\mathscr{H}$.

When applying ELS to an optimization problem, a choice has to be made on which local search procedures to place in the pool. Different pools may lead to different results. When constructing a pool, one may consider how each local search contributes to the overall performance of ELS.

Studies may be available which previously compared and analyzed the performances of different local search procedures on several classes of instances of an optimization problem. For example, Johnson et al. (2002) considered the Asymmetric Traveling Salesman Problem (ATSP). They discussed the performance of a variety of sub-optimal procedures and presented trade-offs between running time and solution quality. Their study included procedures based on local search.

It should be obvious that any local search procedure available for a problem could be placed in the pool; and that ELS will provide a solution at least as good as the solution provided by any individual local search in the pool. While it could be too expensive to include all of the available local search procedures in a pool, it would be adequate to select some subset of these to compose the pool, perhaps based on their known individual performances on different classes of instances. Thus, the question of how to select the local search procedures to form the pool may be answered in part by experimental analyses such as that of Johnson et al. (2002).

Other approaches such as *machine learning* may also be utilized; but the simplest approach to form the pool is *trial-and-error*, thereby placing a certain set of local search procedures in the pool is the *trial* and verifying a poor performance of ELS is associated to the *error*. Thus, one could try some different subsets $\mathscr{I}$ until a subset leads to the desired result.

With ELS, it is necessary to use a method for generating an initial solution to the optimization problem at hand. If a set of initial solutions has been determined, then ELS can be used to improve it.

In this paper, an enhanced local search for the Single Row Facility Layout Problem (SRFLP) is presented and compared with other sub-optimal approaches in the literature. Computational tests with large SRFLP instances with up to 30 departments are also presented.

The structure of this paper is as follows. In the next section the SRFLP is briefly reviewed. In Section 3, an implementation of the enhanced local search is proposed for the SRFLP. The computational results are given in Section 4, followed by the conclusions section.

2. Single Row Facility Layout Problems

In the *Single Row Facility Layout Problem*, a determined number of departments (or machines, offices, workstations, rooms) of a rectangular shape are to be arranged lengthwise on a straight line in a given direction. Consider the following notation:

- n : the number of departments;
- l_i : the length of department i;
- ϕ_{ij} : the material flow transported between department i and department j.

We assume that the parameters n, l_i , ϕ_{ij}, are known. If Π_n denotes the set of all permutations π of $N=\{1, 2,\dots, n\}$. Then, the SRFLP can be mathematically formulated as:

$$\min_{\pi\in\Pi_n} \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \phi_{ij} d_{ij}^{\pi} \tag{1}$$

where d_{ij}^{π} is the distance between the centroids of departments i and j with respect to a permutation π. If the departments in a set $F\subset N$ are placed between i and j in a permutation π, we have: $d_{ij}^{\pi} = \frac{l_i + l_j}{2} + \sum_{k\in F} l_k$. Clearly, when departments i and j are placed side-by-side we have $F=\varnothing$ and, thus, $d_{ij}^{\pi} = \frac{l_i + l_j}{2}$.

The SRFLP is known to be NP-Hard (Beghin-Picavet and Hansen 1982). The problem possesses various applications, for example, supermarket and hospital layouts (Simmons 1969), the arrangement of books on a shelf (Picard and Queyranne 1981) and manufacturing systems design (Heragu and Kusiak 1988).

Among the exact solution approaches presented in the literature are a branch-and-bound algorithm (Simmons 1969, 1971); a dynamic programming approach (Karp and Held 1967; Picard and Queyranne 1981); and linear mixed-integer programming models (Love and Wong 1976; Amaral 2006, 2008). At present, the largest size of problems that could be solved by these exact approaches within reasonable time/memory requirements is n=18. Practical industrial problem sizes can usually reach n = 30, and these sizes are still considered too large to be dealt with using optimal techniques.

Several suboptimal procedures have been proposed for the SRFLP, which are based on: constructive procedures (Hall 1970; Neghabat 1974; Heragu and Kusiak 1988; Kumar et al 1995; Djellab and Gourgand 2001); the penalty method for unconstrained programs (Heragu and Kusiak 1991); semidefinite programming (Anjos et al. 2005); and simulated annealing (Romero and Sánchez-Flores 1990; Heragu and Alfa 1992; Wright et al. 2005).

These procedures have been usually tested on a well-known set of problem instances containing large problem sizes such as n= 20 or n= 30, (See, for example, Kumar et al 1995; Djellab and Gourgand 2001). A lower bound can be used to determine how far a local search solution is from optimality (*optimality gap*). Recently, Anjos and Vannelli (2008) considered large SRFLP instances with up to 30 departments and computed a semidefinite programming based lower bound. For problems with size n=30, very large amounts of time may be required in order to compute the lower bounds. Even though, they allowed the computation to run for several dozen hours and have been able to assert the optimality of feasible solutions for various classical instances.

3. The enhanced local search approach applied to the SRFLP

In order to apply the enhanced local search approach to the SRFLP we need to identify a number of local search procedures, a small subset of which will compose the pool. The local search procedures to be placed in the pool should be chosen so as to try to counterbalance irregularities in individual local search behavior.

3.1 *Local search procedures to the single-row facility layout problem*

Local search procedures for the SRFLP are not difficult to obtain. Consider the following points:

- In a SRFLP with n departments, a solution can be represented by a permutation π of $\{1, \ldots, n\}$. The position of a department in a physical arrangement corresponds to its position in the permutation.
- Each solution is evaluated by a real-valued function f on the set Π_n, given by $f(\pi) = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} c_{ij} d_{ij}^{\pi}$.
- An exchange of the positions of any p departments in a solution is called a *p-opt exchange*. A collection of all possible p-opt exchanges over a solution generates the *p-neighborhood* of that solution (the size of a *p-neighborhood* is $\binom{n}{p}$).

Therefore, different local search procedures can be obtained by specifying the parameter p and how the current solution is iteratively replaced by a neighboring solution.

Definition: A 2-opt exchange that swaps the positions of departments i and j in a permutation π is represented by (i, j). We say that all the possible 2-opt exchanges over a solution are examined in the *natural order* if they are examined in the order:

$(1, 2), \ldots, (1, n), (2, 3), \ldots, (2, n), \ldots, (r, s), \ldots, (n-1, n)$.

The following local search procedures have been in the literature for a long time (for a survey on local search see, for example, Aarts and Lenstra 1997).

3.1.1. *Best Improvement 2-opt* (*BI-2opt*)

In the *Best Improvement 2-opt* (*BI-2opt*) the current solution is to be replaced by the *best* improving solution among all neighboring solutions as shown in Figure 1.

```
Input: An initial solution $\pi$.
Step 1: Find a best solution $\pi'$, according to $f$, in the 2-neighborhood of the current solution $\pi$.
Step 2: If $f(\pi') \geq f(\pi)$ then stop. Otherwise, set $\pi \leftarrow \pi'$ and go to Step 1.
```

Figure 1: The Best Improvement 2-opt (BI-2opt) local search

3.1.2. The First Improvement 2-opt (*FI-2opt*)

In the *First Improvement 2-opt* (*FI-2opt*) local search the neighborhood of the current solution is examined in the natural order. The current solution is replaced by the *first* improving neighboring solution, with respect to the natural order. See Figure 2.

```
Input: An initial solution $\pi$.
Step 1: Let $\pi'$ be the first improving solution that occurs when scanning the 2-neighbohood of $\pi$ in the natural order.
Step 2: If no such $\pi'$ exist, then stop. Otherwise, set $\pi \leftarrow \pi'$ and go to Step 1.
```

Figure 2: The First Improvement 2-opt (FI-2opt) local search

3.1.3. The First Improvement 3-opt (*FI-3opt*)

Similar to the previous local search but we scan the 3-neighborhood of π.

3.2 Selecting local search procedures to place in the pool

The local search procedures presented in the previous sub-section were used to form the following pool: {(*BI-2opt* + *FI-3opt*), (*FI-2opt* + *FI-3opt*), (*FI-3opt* + *BI-2opt*)}, where (*h*1 + *h2*) denotes that the solution provided to local search *h2* was produced by local search *h*1. In preliminary tests (using the trial-and-error method), these local search procedures were chosen, since they formed a set that seemed to lead to the compensatory behavior inherent to the philosophy of ELS.

3.3 Initial solution for the ELS

One can build an initial solution using a construction method or one can simply generate a random initial solution. Here, we take the latter approach.

3.4 The ELS implementation

The proposed ELS implementation is shown in Figure 3. One has to set the parameter *k* (number of initial random solutions provided to ELS). Here, we assume that a local search procedure is never aborted for exceeding a given computing time limit, but in an implementation of ELS, such an early termination could be adopted.

```
Input: $k$ (number of initial random solutions); $\mathcal{J}$ (selected set of local search procedures)
Step 0: Generate $k$ initial solutions at random.
Step 1: Use each of the local search procedures in $\mathcal{J}$ to improve each of the $k$ random solutions.
Step 3: Among the improved solutions obtained, return the best.
```

Figure 3: The ELS implementation

4. Computational experiments with the enhanced local search approach

All the tests were run on a 1.73 GHz PC; the computing time of ELS includes the time taken to read the instance data, and to randomly generate k initial solutions. In this work we use k=10.

For each instance, the ELS implementation (see Figure 3) is run t times because we wish to study the best, average and worst values produced by ELS in these t runs (t was set to 10, similarly to Heragu and Alfa (1992)).

4.1 Comparisons with methods in the literature

The majority of studies about suboptimal procedures for the SRFLP utilize the instances S8H, S10 and S11 of Simmons (1969); instances LW5 and LW11 of Love and Wong (1976); and instances H20 e H30, presented by Heragu and Kusiak (1991). The optimal solutions for these instances have been recently asserted by lower bounds, which required long computing times (Anjos and Vannelli 2008).

Here ELS is compared with other methods in the literature such as the hybrid simulated annealing of Heragu and Alfa (1992); the randomized suboptimal procedures of Kumar et al (1995) and of Djellab and Gourgand (2001).

Heragu and Alfa (1992) present best and averaged results among 10 runs of their hybrid simulated annealing. For the other studies, the best results reported by their authors are used for comparison. Table 1 shows a comparison in terms of solution quality, with optimal values presented in bold face.

Concerning the *best reported results*, it can be seen that the various methods tend to not reach the known optimal value for the largest problem instance, and some failed to reach the (known) optimal value even for smaller problems ($n \leq 20$). Only ELS attained the known optimal solution for all of the seven classical instances of the literature, particularly for the largest instance (n = 30).

Concerning *average results*, the hybrid simulated annealing reached the optimal solution in each of 10 runs for the problems with $n \leq 11$, while ELS reaches the optimal solution in every run for the problems with $n \leq 20$. It is interesting to note that, for the large instance with n = 30, ELS reached the optimal solution in 9 out of 10 runs, and the average (and even the worst) solution reached by ELS is better than the best reported results for the other methods.

Computing times, in seconds, for the several methods are presented in Table 2. Note that different computer platforms have been used in the previous studies and, hence, one cannot compare times directly. Table 2 is used here to provide only an idea of the amounts of time required by the methods on their respective platforms. In 10 runs of ELS on a Pentium PC 1.73 GHz, the largest instance (n=30) requires 220.35 seconds, and the instance n=20 requires 12.92 seconds, on average. For the other six instances, a solution is computed by ELS in less than 1 second on average.

Table 1: Comparison of ELS with methods of the literature on seven classical instances[b].

Problem name	n	Djellab and Gourgand (2001) Best[d]	Kumar et al (1995) Best[d]	Heragu and Alfa (1992) Best[c]	Heragu and Alfa (1992) Average[c]	ELS Best[c]	ELS Average[c]	ELS Worst[c]
LW5	5	**151.0**	**151.0**	**151.0**	**151.0**	**151.0**	**151.0**	**151.0**
S8H	8	**2324.5**	**2324.5**	**2324.5**	**2324.5**	**2,324.5**	**2,324.5**	**2,324.5**
S10	10	**2781.5**	**2781.5**	**2781.5**	**2781.5**	**2,781.5**	**2,781.5**	**2,781.5**
S11	11	**6933.5**	6953.5	**6933.5**	**6933.5**	**6,933.5**	**6,933.5**	**6,933.5**
LW11	11	**6933.5**	7265.5	**6933.5**	**6933.5**	**6,933.5**	**6,933.5**	**6,933.5**
H20	20	**15549.0**	**15549.0**	15602.0	15602.0	**15,549.0**	**15,549.0**	**15,549.0**
H30	30	45019.0	44466.5[a]	45111.0	45111.0	**44,965.0**	44,966.1	44,976.0

a. For the problem with n=30, the lower bound of 44965.0 contradicts the existence of a layout with cost 44466.5 reported by Kumar et al (1995). This is possibly a typo in Kumar's paper; the specific permutation was not reported by Kumar et al and, thus, that entry of the table is unknown (see, Anjos and Vannelli 2008).

b. A local search solution value is presented in bold when it coincides with the lower bound of Anjos and Vannelli 2008 and is, thus, optimal. For the largest instance ELS reached the optimal solution in 9 out of 10 runs. For the remaining problem instances ELS reached the optimal solution in every single run.

c. Taken among 10 runs of the respective method.

d. Best result presented by the authors.

Table 2: Computing times (sec.) required by different methods on their respective platforms.

Problem name	n	Djellab and Gourgand (2001)[b]	Kumar et al (1995)[c]	Heragu and Alfa (1992)[d] Average[a]	ELS[e] Min[a]	ELS[e] Average[a]	ELS[e] Max[a]
LW5	5	0.04	0.01	10.351	0.00	0.01	0.03
S8H	8	0.04	0.08	11.803	0.00	0.04	0.08
S10	10	0.11	0.10	19.815	0.16	0.18	0.19
S11	11	0.37	0.12	23.103	0.27	0.30	0.34
LW11	11	1.11	0.12	29.176	0.25	0.31	0.34
H20	20	4.21	8.60	663.376	11.99	12.92	14.55
H30	30	17.45	91.80	585.947	203.35	220.35	239.56

a. Among 10 runs of the respective method.

b. Djellab and Gourgand (2001) used a SUN SPARC station.

c. Kumar et al. (1995) used a Sun 3/260.

d. Heragu and Alfa (1992) used a mainframe VAX 6420.

e. 1.73 GHz PC.

4.2 Computational tests with new large SRFL instances

Computational tests were also carried out with new large instances of size n= 25 and n= 30 introduced by Anjos and Vannelli (2008). The solution values obtained are presented in Table 3; It can be seen that for one problem ELS reached the optimal solution in 4 out of 10 runs. For the remaining nine problem instances, ELS reached the optimal solution in 8, 9, or all of the 10 runs. The gaps relative to the best, average and worst solution values obtained (see Table 4) reveal that the worst solutions achieved by ELS are indeed very close to the optimum. The maximum gap observed for an average solution was 0.04%.

Table 3: Solution value for 10 runs of ELS on new large instances[a].

Pro-blem name	n	Number of times the optimum was reached[b]	Solution Value Best[b]	Average[b]	Worst[b]
N25_01	25	10	**4,618.0**	**4,618.0**	**4,618.0**
N25_02	25	10	**37,116.5**	**37,116.5**	**37,116.5**
N25_03	25	9	**24,301.0**	24,304.2	24,333.0
N25_04	25	10	**48,291.5**	**48,291.5**	**48,291.5**
N25_05	25	9	**15,623.0**	15,624.6	15,639.0
N30_01	30	10	**8,247.0**	**8,247.0**	**8,247.0**
N30_02	30	9	**21,582.5**	21,583.0	21,587.5
N30_03	30	4	**45,449.0**	45,466.1	45,500.0
N30_04	30	8	**56,873.5**	56,878.5	56,898.5
N30_05	30	9	**115,268.0**	115,268.4	115,272.0

a. A solution value is presented in bold when it coincides with the lower bound of Anjos and Vannelli (2008) and is, thus, optimal.
b. In 10 runs of ELS.

Table 4: GAP of the best, average, worst solutions for 10 runs of ELS on new large instances in the literature.

Pro-blem name	n	Number of times the optimum was reached[b]	GAP (%)[a] Best[b]	Average[b]	Worst[b]
N25_01	25	10	0	0	0
N25_02	25	10	0	0	0
N25_03	25	9	0	0.01	0.13
N25_04	25	10	0	0	0
N25_05	25	9	0	0.01	0.10
N30_01	30	10	0	0	0
N30_02	30	9	0	0	0.02
N30_03	30	4	0	0.04	0.11
N30_04	30	8	0	0.01	0.04
N30_05	30	9	0	0	0
		Maximum:	0	0.04	0.13

a. 100× (Solution value – known optimal value)/ known optimal value.
b. In 10 runs of ELS.

The computing times for the 10 runs of ELS on the new large instances are shown in Table 5. Note that the amount of time taken by ELS is about 1 minute for problems with n=25 and under 4 minutes for problems with n=30, on average, which is very adequate for practical applications.

Table 5: Computing time for 10 runs of ELS on new large instances[a].

Problem name	n	Number of times the optimum was reached[b]	Time (sec.)[a]		
			Min[b]	Average[b]	Max[b]
N25_01	25	10	56.45	63.13	68.44
N25_02	25	10	56.28	63.58	71.84
N25_03	25	9	56.97	61.39	66.63
N25_04	25	10	56.36	63.15	68.27
N25_05	25	9	53.80	60.76	63.97
N30_01	30	10	193.00	206.67	222.19
N30_02	30	9	186.42	214.74	230.44
N30_03	30	4	206.26	219.80	236.72
N30_04	30	8	205.42	222.80	250.03
N30_05	30	9	207.58	227.11	246.44

a. 1.73 GHz PC.
b. In 10 runs of ELS.

5. Conclusions

In this paper, a method called an *enhanced local search* (ELS) was described. The method was applied to the Single Row Facility Layout Problem. Computational experiments showed that ELS quickly attained the (known) optimal solution for all of the classical instances in the literature. The method was also tested on large instances in the literature with 25 and 30 departments and presented a very good performance.

An advantage of ELS is that it admits a direct parallel implementation on parallel computing systems, which may considerably speed up the solution process. Additionally, ELS is a general method and can be applied to other combinatorial optimization problems.

Acknowledgements

This work was supported by CNPq (500212/2002-3).

References

Aarts E., Lenstra J. K. (Eds.) 1997. Local Search in Combinatorial Optimization.Wiley, Chichester, UK.

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

Amaral, A.R.S., 2008. An exact approach for the one-dimensional facility layout problem, Operations Research, to appear.

Anjos, M.F., Kennings A., Vannelli, A. 2005. A Semidefinite Optimization Approach for the Single-Row Layout Problem with Unequal Dimensions. Discrete Optimization, 2, 113–122.

Anjos, M.F., Vannelli, A. 2008. Globally optimal solutions for large single-row facility layout problems. Informs Journal on Computing, to appear.

Beghin-Picavet, M., Hansen, P. 1982. Deux problemes d'affectation non lineaires. RAIRO Recherche Opérationelle, 16, 263–276.

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

Hall, K. M. 1970. An r-dimensional quadratic placement algorithm. Management Science, 17, 219–229.

Heragu, S. S., Kusiak, A. 1988. Machine layout problem in flexible manufacturing systems. Operations Research, 36, 258–268.

Heragu, S. S., Kusiak, A. 1991. Efficient models for the facility layout problem. European Journal of Operational Research, 53, 1–13.

Heragu, S. S., Alfa, A. S. 1992. Experiment Analysis of simulated annealing based algorithms for the layout problem. European Journal of Operational Research, 57, 190–202.

Johnson, D. S., Gutin, G., McGeoch, L. A., Yeo, A., Zhang, W., & Zverovitch, A. 2002. The Traveling Salesman Problem and its Variations, chap. Experimental Analysis of Heuristics for the ATSP, pp. 445–487. Kluwer Academic Publishers, Dordrecht.

Karp, R.M., Held, M. 1967. Finite-state processes and dynamic programming, SIAM Journal of Applied Mathematics, 15, 693–718.

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

Love, R. F., Wong, J. Y. 1976. On solving a one-dimensional space allocation problem with integer programming. INFOR, 14, 139–143.

Neghabat, F. 1974. An efficient equipment layout algorithm. Operations Research, 22, 622–628.

Picard, J., Queyranne, M. 1981. On the one dimensional space allocation problem. Operations Research, 29, 371–391.

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

Simmons, D. M. 1969. One-dimensional space allocation: an ordering algorithm. Operations Research, 17, 812–826. Simmons, D. M. 1971. A further note on one-dimensional space allocation. Operations Research, 19, 249.

Wright, M.B., Amaral, A.R.S., Camatta, A. 2005. Subset moves based simulated annealing applied to the layout problem. Presented at the 6th Metaheuristics International Conference, Vienna.