INFORMS Journal on Computing
Vol. 20, No. 4, Fall 2008, pp. 611–617
ISSN 1091-9856 | EISSN 1526-5528 | 08 | 2004 | 0611

DOI 10.1287/ijoc.1080.0270

Computing Globally Optimal Solutions for Single-Row Layout Problems Using Semidefinite Programming and Cutting Planes

Miguel F. Anjos
Department of Management Sciences, University of Waterloo, Waterloo, Ontario N2L 3G1, Canada, anjos@stanfordalumni.org

Anthony Vannelli
School of Engineering, University of Guelph, Guelph, Ontario N1G 2W1, Canada, vannelli@uoguelph.ca

This paper is concerned with the single-row facility layout problem (SRFLP). A globally optimal solution to the SRFLP is a linear placement of rectangular facilities with varying lengths that achieves the minimum total cost associated with the (known or projected) interactions between them. We demonstrate that the combination of a semidefinite programming relaxation with cutting planes is able to compute globally optimal layouts for large SRFLPs with up to 30 facilities. In particular, we report the globally optimal solutions for two sets of SRFLPs previously studied in the literature, some of which have remained unsolved since 1988.

Key words: single-row facility layout; space allocation; semidefinite programming; cutting planes; combinatorial optimization
History: Accepted by Karen Aardal, Area Editor for Design and Analysis of Algorithms; received February 2007; revised August 2007; accepted January 2008. Published online in *Articles in Advance* May 15, 2008.

1. Introduction

The single-row facility layout problem (SRFLP) is concerned with the arrangement of a given number of rectangular facilities next to each other along a line so as to minimize the total weighted sum of the center-to-center distances between all pairs of facilities. This problem is a special case of the unequal-area facility layout problem and is also known in the literature as the one-dimensional space allocation problem; see, e.g., Picard and Queyranne (1981). An instance of the SRFLP consists of n one-dimensional facilities, denoted $1, \ldots, n$, with given positive lengths $l_1, \ldots, l_n$, and pairwise weights c_{ij}. The objective is to arrange the facilities so as to minimize the total weighted sum of the center-to-center distances between all pairs of facilities. If all the facilities have the same length, the SRFLP becomes an instance of the linear ordering (or linear arrangement) problem; see, e.g., Liu and Vannelli (1995) and Mitchell and Borchers (2000), which is itself a special case of the quadratic assignment problem; see, e.g., Çela (1998). Several applications of the SRFLP have been identified in the literature. One such application arises in the area of flexible manufacturing systems, where machines within manufacturing cells are often placed along a straight path travelled by an automated guided vehicle; see, e.g., Heragu and Kusiak (1988).

The SRFLP was first studied by Simmons (1969), who proposed a branch-and-bound algorithm. Subsequently, Picard and Queyranne (1981) developed a dynamic programming algorithm. Mixed-integer linear programming models have also been proposed; see, e.g., Grötschel et al. (1984), Reinelt (1985), and most recently, Amaral (2006). Although these algorithms are guaranteed to find the global optimal solution, they have very high computational time and memory requirements and are unlikely to be effective for problems with more than about 20 facilities. Several heuristic algorithms for the SRFLP have also been proposed. We point out the application of nonlinear optimization methods by Heragu and Kusiak (1991), the simulated annealing algorithms proposed independently by Romero and Sánchez-Flores (1990) and Heragu and Alfa (1992), a greedy heuristic algorithm of Kumar et al. (1995), and the more recent metaheuristics proposed by de Alvarenga et al. (2000). However, none of these algorithms provides a guarantee of global optimality or an estimate of the distance from optimality. Progress in obtaining such estimates was reported in Anjos et al. (2005), where tight global lower bounds were obtained using a semidefinite programming relaxation.

Semidefinite programming (SDP) refers to the class of optimization problems where a linear function of a symmetric matrix variable X is optimized subject to linear constraints on the elements of X and the additional constraint that X must be positive semidefinite. This includes linear programming problems as a special case, namely, when the matrix variable is diagonal. A variety of algorithms for solving SDP problems, including polynomial-time interior-point algorithms, have been implemented and benchmarked, and several excellent solvers for SDP are now available. We refer the reader to the books by de Klerk (2002) and Wolkowicz et al. (2000) for a thorough coverage of the theory and algorithms in this area, as well as of several application areas where SDP researchers have made significant contributions. In particular, SDP has been very successfully applied to problems, which, like the SRFLP, have a combinatorial nature. Recent survey papers on the application of SDP to combinatorial optimization include Anjos (2005), Anjos and Wolkowicz (2002), and Laurent and Rendl (2005).

Anjos et al. (2005) proposed an SDP relaxation of the SRFLP as well as a heuristic to extract a linear placement from the optimal matrix solution to the SDP relaxation. Therefore, this relaxation yields both a feasible solution to the given SRFLP instance and a guarantee of how far it is from global optimality. The results reported in Anjos et al. (2005) show that this approach yields layouts that are consistently a few percentage points from global optimality for randomly generated instances with up to 80 facilities. More recently, Anjos and Vannelli (2006) experimented with a branch-and-bound algorithm for the SRFLP that solves the SDP relaxation at each node. Although this approach yielded solutions that are provably very close to global optimality (typically less than 1% gap) for randomly generated instances of the SRFLP with up to 40 facilities with a reasonable amount of computational effort, the results also suggest that branching does not provide a significant improvement over the results obtained at the root node of the branch-and-bound tree.

In this paper, we explore the approach of tightening the SDP relaxation by introducing cutting planes at the root node only. Our computational results show that a straightforward implementation of this approach using only the so-called triangle inequalities is highly effective. In particular, we obtain the first globally optimal layouts for large SRFLPs with up to 30 facilities, some of which have been studied in the literature since 1988 but have remained unsolved.

This paper is structured as follows. In §2, we revisit the new formulation of the SRFLP proposed in Anjos et al. (2005) and the corresponding SDP relaxation. In §3, we introduce the cutting planes that are used to strengthen the relaxation. In §4, we briefly explain the implementation of this SDP-based approach and report our computational results.

2. A Quadratic Formulation of the SRFLP and Its SDP Relaxation

Let $\pi = (\pi_1, \ldots, \pi_n)$ denote a permutation of the indices $[n] := \{1, 2, \ldots, n\}$ of the facilities, so that the leftmost facility is π_1, the facility to the right of it is π_2, and so on, with π_n being the last facility in the arrangement. Given a permutation π and two distinct facilities i and j, the center-to-center distance between i and j with respect to this permutation is $\frac{1}{2}l_i + D_\pi(i,j) + \frac{1}{2}l_j$, where $D_\pi(i,j)$ denotes the sum of the lengths of the facilities between i and j in the ordering defined by π. To solve the SRFLP, we seek a permutation of the facilities that minimizes the weighted sum of the distances between all pairs of facilities. We express this objective as

$$\min_{\pi\in\Pi_n} \sum_{i<j} c_{ij}\left[\tfrac{1}{2}l_i + D_\pi(i,j) + \tfrac{1}{2}l_j\right],$$

where Π_n denotes the set of all permutations of $[n]$.

Simmons (1969) observed that if we rewrite the objective function as

$$\min_{\pi\in\Pi_n} \sum_{i<j} c_{ij} D_\pi(i,j) + \sum_{i<j} \tfrac{1}{2}c_{ij}(l_i + l_j),$$

where the second summation is a constant independent of π, then it is clear that the crux of the problem is to minimize $\sum_{i<j} c_{ij} D_\pi(i,j)$ over all permutations π. Furthermore, it is clear that $D_\pi(i,j) = D_{\pi'}(i,j)$, where π' denotes the permutation symmetric to π, defined by $\pi'_i = \pi_{n+1-i}$, $i = 1, \ldots, n$. This shows that we can exchange the left and right ends of the layout and obtain the same objective value. Hence, it is possible to simplify the problem by considering only the permutations for which, say, facility 1 is in the left half of the arrangement. This type of symmetry-breaking strategy is important for reducing the computational requirements of most algorithms, including those based on linear programming or dynamic programming. One noteworthy aspect of the SDP-based approach is that it implicitly accounts for these symmetries and thus does not require the use of additional explicit symmetry-breaking constraints.

The SDP relaxation for the SRFLP proposed in Anjos et al. (2005) is obtained as follows. Define a binary ± 1 variable for each pair i, j of facilities with $i < j$ such that

$$R_{ij} := \begin{cases} 1 & \text{if facility } i \text{ is to the right of facility } j, \\ -1 & \text{if facility } i \text{ is to the left of facility } j. \end{cases}$$

In this definition, the order of the subscripts matters, and $R_{ij} = -R_{ji}$. Thus, the R_{ij} variables have $i < j$.

To express the objective function of the SRFLP in terms of the variables R_{ij}, it suffices to observe that because k is between i and j if and only if $R_{ki}R_{kj} = -1$, the sum of the lengths of the facilities between i and j can be expressed as

$$\sum_{k\neq i,\,j} l_k\left(\frac{1-R_{ki}R_{kj}}{2}\right).$$

To accurately formulate the SRFLP, it is further required that the R_{ij} variables represent a valid arrangement of the n facilities. Therefore, we require that if $R_{ij} = R_{jk}$, then $R_{ik} = R_{ij}$, a necessary transitivity condition that can be formulated as $\binom{n}{3}$ quadratic constraints:

$$R_{ij}R_{jk} - R_{ij}R_{ik} - R_{ik}R_{jk} = -1 \quad \text{for all triples } i<j<k. \tag{1}$$

This leads to the following formulation of the SRFLP:

$$\begin{aligned} \min\;& K-\sum_{i<j}\frac{c_{ij}}{2}\Bigg[\sum_{k<i} l_k R_{ki}R_{kj} - \sum_{i<k<j} l_k R_{ik}R_{kj} + \sum_{k>j} l_k R_{ik}R_{jk}\Bigg] \\ \text{s.t.}\;& R_{ij}R_{jk} - R_{ij}R_{ik} - R_{ik}R_{jk} = -1 \quad \text{for all triples } i<j<k, \\ & R_{ij}^2 = 1 \quad \text{for all } i<j, \end{aligned} \tag{2}$$

where $K := (\sum_{i<j}(c_{ij}/2))(\sum_{k=1}^{n} l_k)$. Note that if every R_{ij} variable is replaced by its negative, then there is no change whatsoever to the formulation. This is how our formulation, and the subsequent SDP relaxation, implicitly take into account the natural symmetry of the SRFLP.

We can now formulate the SRFLP in the space of real symmetric matrices. Fixing an ordering of all pairs ij such that $i<j$, we define the vector

$$v := \left(R_{p_1}, \ldots, R_{p_{\binom{n}{2}}}\right)^T,$$

where p_k denotes the kth pair ij in the ordering. Using v, we construct the rank-one matrix $X := vv^T$, whose rows and columns are indexed by pairs ij. By construction, $X_{p_i,p_j} = R_{p_i}R_{p_j}$ for any two pairs p_i, p_j, and therefore we can formulate the SRFLP as

$$\begin{aligned} \min\;& K-\sum_{i<j}\frac{c_{ij}}{2}\Bigg[\sum_{k<i} l_k X_{ki,\,kj} - \sum_{i<k<j} l_k X_{ik,\,kj} + \sum_{k>j} l_k X_{ik,\,jk}\Bigg] \\ \text{s.t.}\;& X_{ij,\,jk} - X_{ij,\,ik} - X_{ik,\,jk} = -1 \quad \text{for all triples } i<j<k, \\ & \operatorname{diag}(X) = e, \\ & \operatorname{rank}(X) = 1, \\ & X \succeq 0, \end{aligned} \tag{3}$$

where $\operatorname{diag}(X)$ represents a vector containing the diagonal elements of X, e denotes the vector of all ones, and $X \succeq 0$ denotes that X is symmetric positive semidefinite (see Anjos et al. 2005 for more details). Removing the rank constraint yields the SDP relaxation. Note that in general the SDP problem provides only a lower bound on the optimal value of the SRFLP, and not a feasible solution to (2), unless the optimal matrix X^* happens to have rank equal to one.

Before proceeding, we observe that the formulation and SDP relaxation above are closely related to the basic SDP relaxation for the max-cut problem used in the groundbreaking paper of Goemans and Williamson (1995). The max-cut SDP relaxation can be interpreted as a relaxation of the so-called cut polytope, an important and well-known structure in the area of integer programming. The reader is referred to Deza and Laurent (1997) for a wealth of results about the cut polytope. In particular, the cutting planes that we use in §3 are well-known facets of the cut polytope.

Although the SDP relaxation shares some common structure with the max-cut relaxation for which Goemans and Williamson (1995) analyzed a well-known randomized rounding procedure, we cannot use that procedure here because it does not ensure that the equality constraints (1) hold. That is why Anjos et al. (2005) devised a different procedure to obtain a permutation from the optimal solution to the SDP relaxation. The procedure is as follows: If X^* is the optimal solution to the SDP relaxation, then each row of X^* corresponds to a specific pair $i_1 j_1$ of facilities. Therefore, for any row of X^*, if we set $R_{i_1 j_1} = +1$, then we can scan the other entries of the row and assign the value $X_{i_1 j_1,\, i_2 j_2}$ to the variable $R_{i_2 j_2}$ for every pair $i_2 j_2 \neq i_1 j_1$. Using these values, we compute

$$\omega_k = \tfrac{1}{2}\Bigg(n+1+\sum_{j\neq k} R_{kj}\Bigg)$$

for $k = 1, \ldots, n$. The motivation for the values ω_k comes from the fact proved in Anjos et al. (2005) that if X^* is rank-one, then the values ω_k, $k = 1, \ldots, n$ are all distinct and belong to $[n]$, and hence define a permutation of $[n]$. In general, $\operatorname{rank}(X^*) > 1$ and thus $\omega_k \in [1, n]$, so the SDP-based heuristic obtains a permutation of $[n]$ by sorting the values ω_k. The sorting can be in either decreasing or increasing order (because the objective value is the same), and because the procedure implicitly sets $R_{i_1 j_1} = +1$, we choose the order that places i_1 to the right of j_1. The output of the heuristic is the best layout found by considering every row in turn.

3. Semidefinite Relaxation and Cutting Planes

Our objective is to tighten the SDP relaxation:

$$\begin{aligned}
\min\ & K-\sum_{i<j}\frac{c_{ij}}{2}\left[\sum_{k<i} l_k X_{ki,kj}-\sum_{i<k<j} l_k X_{ik,kj}+\sum_{k>j} l_k X_{ik,jk}\right]\\
\text{s.t. } & X_{ij,jk}-X_{ij,ik}-X_{ik,jk}=-1\\
&\qquad\qquad \text{for all triples } i<j<k,\\
& \operatorname{diag}(X)=e,\\
& X\succeq 0.
\end{aligned}\tag{4}$$

A standard way to tighten linear or semidefinite relaxations of integer optimization problems is to add inequalities that are valid for the integer feasible points. There are several classes of such inequalities that can be considered. We consider only the so-called triangle inequalities, a well-known class of valid inequalities for the cut polytope. These inequalities model the fact that for any assignment of ± 1 to the entries of X, the entries X_{p_1,p_2}, X_{p_1,p_3}, and X_{p_2,p_3}, where p_1, p_2, and p_3 are any three distinct pairs, must comprise an even number of negative ones. Indeed, by virtue of integrality, every feasible solution of (2) satisfies the following $\binom{\binom{n}{2}}{3}$ inequalities:

$$\begin{aligned}
R_{p_1}R_{p_2}+R_{p_1}R_{p_3}+R_{p_2}R_{p_3}&\geq -1,\\
R_{p_1}R_{p_2}-R_{p_1}R_{p_3}-R_{p_2}R_{p_3}&\geq -1,\\
-R_{p_1}R_{p_2}-R_{p_1}R_{p_3}+R_{p_2}R_{p_3}&\geq -1,\\
-R_{p_1}R_{p_2}+R_{p_1}R_{p_3}-R_{p_2}R_{p_3}&\geq -1,
\end{aligned}$$

for every triple of pairs p_1, p_2, p_3. In terms of the matrix representation of (3), we have

$$\begin{aligned}
X_{p_1,p_2}+X_{p_1,p_3}+X_{p_2,p_3}&\geq -1,\\
X_{p_1,p_2}-X_{p_1,p_3}-X_{p_2,p_3}&\geq -1,\\
-X_{p_1,p_2}-X_{p_1,p_3}+X_{p_2,p_3}&\geq -1,\\
-X_{p_1,p_2}+X_{p_1,p_3}-X_{p_2,p_3}&\geq -1.
\end{aligned}$$

Although these hold for X feasible for (3), they will not (in general) hold for X feasible for (4).

However, it turns out that some of these inequalities do hold for all the feasible matrices of (4). Indeed, if X is feasible for (4), then for every triple of pairs i_1i_2, i_1i_3, and i_2i_3, where $i_1<i_2<i_3$, the entries of X automatically satisfy

$$\begin{aligned}
X_{i_1i_2,i_1i_3}+X_{i_1i_2,i_2i_3}+X_{i_1i_3,i_2i_3}&\geq -1,\\
X_{i_1i_2,i_1i_3}-X_{i_1i_2,i_2i_3}-X_{i_1i_3,i_2i_3}&\geq -1,\\
-X_{i_1i_2,i_1i_3}-X_{i_1i_2,i_2i_3}+X_{i_1i_3,i_2i_3}&\geq -1,\\
-X_{i_1i_2,i_1i_3}+X_{i_1i_2,i_2i_3}-X_{i_1i_3,i_2i_3}&\geq -1.
\end{aligned}$$

Because $X_{i_1i_2,i_2i_3}-X_{i_1i_2,i_1i_3}-X_{i_1i_3,i_2i_3}=-1$, the fourth inequality trivially holds. For the first inequality, note that

$$\begin{aligned}
&X_{i_1i_2,i_2i_3}+X_{i_1i_2,i_1i_3}+X_{i_1i_3,i_2i_3}\\
&\quad = X_{i_1i_2,i_2i_3}+X_{i_1i_2,i_1i_3}+(1+X_{i_1i_2,i_2i_3}-X_{i_1i_2,i_1i_3})\\
&\quad = 1+2X_{i_1i_2,i_2i_3}\\
&\quad \geq -1
\end{aligned}$$

because $X\succeq 0$ and $\operatorname{diag}(X)=e$ together imply $|X_{i_1i_2,i_2i_3}|\leq 1$. Similar arguments show that the remaining two inequalities also hold. Therefore, $4\binom{n}{3}$ triangle inequalities automatically hold for all the feasible matrices of (4).

It is a natural approach to improve the relaxation by adding to it all the triangle inequalities that are not automatically enforced. If we add the remaining triangle inequalities, we obtain the following (tighter) relaxation:

$$\begin{aligned}
\min\ & K-\sum_{i<j}\frac{c_{ij}}{2}\left[\sum_{k<i} l_k X_{ki,kj}-\sum_{i<k<j} l_k X_{ik,kj}+\sum_{k>j} l_k X_{ik,jk}\right]\\
\text{s.t. } & X_{i_1i_2,i_1i_3}-X_{i_1i_2,i_2i_3}-X_{i_1i_3,i_2i_3}=-1\\
&\qquad\qquad \forall\, 1\leq i_1<i_2<i_3\leq n,\\
& X_{p_1,p_2}+X_{p_1,p_3}+X_{p_2,p_3}\geq -1,\\
& X_{p_1,p_2}-X_{p_1,p_3}-X_{p_2,p_3}\geq -1,\\
& -X_{p_1,p_2}-X_{p_1,p_3}+X_{p_2,p_3}\geq -1,\\
& -X_{p_1,p_2}+X_{p_1,p_3}-X_{p_2,p_3}\geq -1,\\
&\qquad \forall\, p_1,p_2,p_3 \text{ not with the form above}\\
& \operatorname{diag}(X)=e,\\
& X\succeq 0.
\end{aligned}\tag{5}$$

The relaxation (4) has $O(n^3)$ linear equality constraints, and the relaxation (5) has the same number of

Table 1 **Globally Optimal Solutions for Five Well-Known Test Problems**

Instance	n	Optimal value of (4)	CPU time (m:s)	Best layout by SDP-based heuristic using X^* of (4)	Gap (%)	Improved SDP bound	Best layout by SDP-based heuristic	Gap (%)
Lit-1	8	2,324.5	0:00.2	2,324.5	0	2,324.5	2,324.5	0
Lit-2	10	2,773.9	0:00.3	2,781.5	0.27	2,781.5	2,781.5	0
Lit-3	11	6,847.6	0:00.3	6,933.5	1.24	6,933.5	6,933.5	0
Lit-4	20	15,285.9	0:10.7	15,549.0	1.69	15,549.0	15,549.0	0
Lit-5	30	43,963.7	3:17.0	45,115.0	2.64	44,965.0	44,965.0	0

Table 2 Computational Cost to Find the Global Optimal Solutions

Instance	CPU time (h:m:s)	Number of rounds of cut generation	Total number of cuts added	CPU time when the SDP-based heuristic found global optimum
Lit-1	0:00:00.2	0	0	0:00:00.2
Lit-2	0:00:03.4	1	342	0:00:01.4
Lit-3	0:00:32.6	3	1,051	0:00:01.9
Lit-4	0:26:53.5	8	2,871	0:00:08.4
Lit-5	15:50:57.0	20	6,770	10:40:20.4

equality constraints, plus $O(n^6)$ inequality constraints. Obviously, in practice, the triangle inequalities cannot all be included simultaneously (except perhaps for very small values of n). A practical approach to problems such as (5) is to begin by solving (4), then add some violated triangle inequalities, reoptimize, and repeat until no more triangle inequalities are violated. This is the approach we used to obtain the computational results in the next section.

4. Computational Results

In this section, we show that using the relaxation (4) augmented with a few hundred triangle inequalities, we can obtain globally optimal layouts for SRFLPs with up to 30 facilities. The computational properties of the relaxation (4), and in particular its ability to yield tight bounds for SRFLPs, have already been studied by Anjos et al. (2005) and Anjos and Vannelli (2006). Therefore, we focus here on the effect of adding triangle inequalities to (4). All the computational results were obtained on a 2.0 GHz Dual Opteron with 16 GB of RAM, and the SDP problems were solved using the interior-point solver CSDP (version 5.0) of Borchers (1999) in conjunction with the ATLAS library of routines of Whaley et al. (2001).

Our approach begins by solving the relaxation (4). Next, we apply the SDP-based heuristic described in §2 to obtain specific permutations and update the best permutation found so far. Then, we sort the triangle inequalities in terms of their violation at the current matrix solution and choose a cutoff value for the violations that yields the 300 to 400 most-violated inequalities. We add these most-violated inequalities to the relaxation, reoptimize, reapply the heuristic, and update the best permutation found. This process is repeated until no more triangle inequalities are violated. More sophisticated techniques could certainly be used and might improve the performance of the SDP-based approach. In spite of its simplicity, this approach gives excellent results.

Table 4 Computational Cost to Find the Global Optimal Solutions

Instance	CPU time (h:m:s)	Number of rounds of cut generation	Total number of cuts added	CPU time when the SDP-based heuristic found global optimum
Lit-Cl-1	0:00:00.1	0	0	0:00:00.1
Lit-Cl-2	0:00:00.4	0	0	0:00:00.3
Lit-Cl-3	0:00:01.2	1	336	0:00:00.3
Lit-Cl-4	0:00:01.8	1	320	0:00:00.4
Lit-Cl-5	0:00:32.8	3	989	0:00:00.4
Lit-Cl-6	0:05:52.6	6	2,115	0:00:01.8
Lit-Cl-7	0:41:23.3	10	3,317	0:11:26.4
Lit-Cl-8	51:06:52.7	31	10,584	25:25:48.9

4.1. Optimal Solutions for Five Well-Known Instances

First, we report globally optimal solutions for five well-known test problems from the literature. The first three problems come from Simmons (1969), while the larger two problems were first considered in Heragu and Kusiak (1988).

We point out that the global optima in Tables 1 and 2 were the best solutions found by the metaheuristics of de Alvarenga et al. (2000), but with no proof (or claim) of global optimality. We also observe that for instance Lit-5, a layout of cost 44,466.5 was reported in Kumar et al. (1995), which our lower bound contradicts. We suppose that it is a typo and that the correct figure was 44,966.5, but the specific permutation was not reported by Kumar et al. (1995).

4.2. Optimal Solutions for Eight Instances with Clearance Requirement

The next eight instances were also introduced for the first time in Heragu and Kusiak (1988). They

Table 3 Globally Optimal Solutions for Eight Instances with Clearance Requirement

Instance	n	Best layout prev. known	Optimal value of (4)	CPU time (m:s)	Best layout by SDP-based heuristic using X^* of (4)	Gap (%)	Improved SDP bound	Best layout by SDP-based heuristic	Gap (%)
Lit-Cl-1	5	1.100	1.100	0:00.1	1.100	0	1.100	1.100	0
Lit-Cl-2	6	1.990	1.990	0:00.4	1.990	0	1.990	1.990	0
Lit-Cl-3	7	4.730	4.659	0:00.4	4.730	1.50	4.730	4.730	0
Lit-Cl-4	8	6.295	6.169	0:00.6	6.295	2.00	6.295	6.295	0
Lit-Cl-5	12	24.675	22.667	0:04.1	23.365	2.99	23.365	23.365	0
Lit-Cl-6	15	49.375	43.998	0:17.8	44.600	1.35	44.600	44.600	0
Lit-Cl-7	20	141.040	117.454	1:41.8	120.220	2.30	119.710	119.710	0
Lit-Cl-8	30	395.770	326.587	3:14.4	336.440	2.93	334.870	334.870	0

Table 5 Globally Optimal Solutions for 10 New Large Instances

Instance	n	Optimal value of (4)	CPU time (m:s)	Best layout by SDP-based heuristic using X^* of (4)	Gap (%)	Improved SDP bound	Best layout by SDP-based heuristic	Gap (%)
Nugent25-01	25	4,514.7	0:44.1	4,622.0	2.32	4,618.0	4,618.0	0
Nugent25-02	25	36,355.4	0:44.8	37,641.5	3.42	37,116.5	37,116.5	0
Nugent25-03	25	23,690.6	0:44.4	24,537.0	3.45	24,301.0	24,301.0	0
Nugent25-04	25	47,329.8	0:43.2	48,887.5	3.19	48,291.5	48,291.5	0
Nugent25-05	25	15,304.1	0:43.8	15,767.0	2.94	15,623.0	15,623.0	0
Nugent30-01	30	8,060.8	3:15.8	8,305.0	2.94	8,247.0	8,247.0	0
Nugent30-02	30	21,188.1	3:04.7	21,663.5	2.19	21,582.5	21,582.5	0
Nugent30-03	30	44,518.5	3:06.0	45,712.0	2.61	45,449.0	45,449.0	0
Nugent30-04	30	55,947.2	3:09.3	56,922.5	1.71	56,873.5	56,873.5	0
Nugent30-05	30	113,071.7	3:10.0	115,776.0	2.34	115,268.0	115,268.0	0

differ from the previous instances in that a clearance requirement of 0.01 unit length between each pair of consecutive facilities is required. This requirement is motivated by the context of an application in flexible manufacturing systems. Because the required clearance is always the same, it is straightforward to account for this requirement in our model by appropriately adjusting the lengths of every facility. Using our approach, we obtained globally optimal solutions for the eight instances. Note in Tables 3 and 4 that for the four largest instances, the global optima are *strict improvements over the best layouts previously known* (Kumar et al. 1995).

4.3. Optimal and Near-Optimal Solutions for Large New Instances

Finally, to further demonstrate the performance of the SDP-based algorithm, we generated a number of new instances of the SRFLP by starting with the connectivity data from the well-known Nugent quadratic assignment problems with 25 and 30 facilities (originally studied in Nugent et al. 1968) and adding to them randomly generated facility lengths (see Tables 5 and 6).

Table 6 Computational Cost to Find the Global Optimal Solutions

Instance	CPU time (h:m:s)	Number of rounds of cut generation	Total number of cuts added	CPU time when the SDP-based heuristic found global optimum
Nugent25-01	3:44:38.4	14	4,798	0:32:00.9
Nugent25-02	4:50:27.4	16	5,484	1:11:09.1
Nugent25-03	5:48:21.0	17	5,907	3:29:27.5
Nugent25-04	4:04:51.1	15	5,255	1:29:25.4
Nugent25-05	8:22:21.8	19	6,611	5:04:19.3
Nugent30-01	7:41:06.1	14	4,924	3:08:37.9
Nugent30-02	10:41:53.1	17	5,809	5:00:47.6
Nugent30-03	19:32:01.0	21	7,297	11:43:26.3
Nugent30-04	31:03:10.5	24	8,587	1:35:09.7
Nugent30-05	19:54:06.6	22	7,294	7:51:58.2

5. Conclusions

We demonstrated that the combination of a semidefinite programming relaxation with cutting planes is able to compute globally optimal layouts for large SRFLPs with up to 30 facilities. Our computational results suggest that this approach can routinely obtain optimal layouts for SRFLPs with up to 25 facilities in a few hours, and in several dozen hours for SRFLPs with up to 30 facilities. In particular, we reported the globally optimal solutions for two sets of SRFLPs previously studied in the literature, some of which had remained unsolved since 1988.

Acknowledgments

The authors thank Robert J. Vanderbei for suggesting the extension of the heuristic from Anjos et al. (2005) that is presented in §2. They also thank C. Helmberg, F. Rendl, and R. J. Vanderbei for their encouraging and helpful comments on this line of research, and B. Borchers for assisting in installing the CSDP and ATLAS software used to obtain the computational results. This research was partially supported by Grants 312125, 314668, and 15296 from the Natural Sciences and Engineering Research Council of Canada, and by a Bell University Laboratories Research Grant.

References

Amaral, A. R. S. 2006. On the exact solution of a facility layout problem. *Eur. J. Oper. Res.* **173** 508–518.

Anjos, M. F. 2005. Semidefinite optimization approaches for satisfiability and maximum-satisfiability problems. *J. Satisfiability, Boolean Model. Comput.* **1** 1–47.

Anjos, M. F., A. Vannelli. 2006. On the computational performance of a semidefinite programming approach to single row layout problems. *Proc. Oper. Res. 2005*, Springer-Verlag, Berlin, 277–282.

Anjos, M. F., H. Wolkowicz. 2002. Semidefinite programming for discrete optimization and matrix completion problems. *Discrete Appl. Math.* **123** 513–577.

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

Borchers, B. 1999. CSDP, a C library for semidefinite programming. *Optim. Methods Software* **11** 613–623.

Çela, E. 1998. *The Quadratic Assignment Problem, Combinatorial Optimization*, Vol. 1. Kluwer Academic Publishers, Dordrecht, The Netherlands.

de Alvarenga, A. G., F. J. Negreiros-Gomes, M. Mestria. 2000. Metaheuristic methods for a class of the facility layout problem. *J. Intell. Manuf.* **11** 421–430.

de Klerk, E. 2002. *Aspects of Semidefinite Programming, Applied Optimization*, Vol. 65. Kluwer Academic Publishers, Dordrecht, The Netherlands.

Deza, M. M., M. Laurent. 1997. *Geometry of Cuts and Metrics, Algorithms and Combinatorics*, Vol. 15. Springer-Verlag, Berlin.

Goemans, M. X., D. P. Williamson. 1995. Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *J. Assoc. Comput. Mach.* **42** 1115–1145.

Grötschel, M., M. Jünger, G. Reinelt. 1984. A cutting plane algorithm for the linear ordering problem. *Oper. Res.* **32** 1195–1220.

Heragu, S. S., A. S. Alfa. 1992. Experimental analysis of simulated annealing based algorithms for the layout problem. *Eur. J. Oper. Res.* **57** 190–202.

Heragu, S. S., A. Kusiak. 1988. Machine layout problem in flexible manufacturing systems. *Oper. Res.* **36** 258–268.

Heragu, S. S., A. Kusiak. 1991. Efficient models for the facility layout problem. *Eur. J. Oper. Res.* **53** 1–13.

Kumar, K. R., G. C. Hadjinicola, T. Lin. 1995. A heuristic procedure for the single-row facility layout problem. *Eur. J. Oper. Res.* **87** 65–73.

Laurent, M., F. Rendl. 2005. Semidefinite programming and integer programming. K. Aardal, G. Nemhauser, R. Weismantel, eds. *Handbook on Discrete Optimization*. Elsevier, Amsterdam, 393–514.

Liu, W., A. Vannelli. 1995. Generating lower bounds for the linear arrangement problem. *Discrete Appl. Math.* **59** 137–151.

Mitchell, J. E., B. Borchers. 2000. Solving linear ordering problems with a combined interior point/simplex cutting plane algorithm. H. Frenk, K. Roos, T. Terlaky, S. Zhang, eds. *High Performance Optimization*, Vol. 33. Kluwer Academic Publishers, Dordrecht, The Netherlands, 349–366.

Nugent, C. E., T. E. Vollmann, J. Ruml. 1968. An experimental comparison of techniques for the assignment of facilities to locations. *Oper. Res.* **16** 150–173.

Picard, J.-C., M. Queyranne. 1981. On the one-dimensional space allocation problem. *Oper. Res.* **29** 371–391.

Reinelt, G. 1985. *The Linear Ordering Problem: Algorithms and Applications, Research and Exposition in Mathematics*, Vol. 8. Heldermann Verlag, Berlin.

Romero, D., A. Sánchez-Flores. 1990. Methods for the one-dimensional space allocation problem. *Comput. Oper. Res.* **17** 465–473.

Simmons, D. M. 1969. One-dimensional space allocation: An ordering algorithm. *Oper. Res.* **17** 812–826.

Whaley, R., A. Petitet, J. Dongarra. 2001. Automated empirical optimizations of software and the ATLAS project. *Parallel Comput.* **27** 3–35.

Wolkowicz, H., R. Saigal, L. Vandenberghe, eds. 2000. *Handbook of Semidefinite Programming*. Kluwer Academic Publishers, Boston.