On the Computational Performance of a Semidefinite Programming Approach to Single Row Layout Problems *

Miguel F. Anjos[1] and Anthony Vannelli[2]

[1] Department of Management Sciences, University of Waterloo, Waterloo, Ontario, Canada N2L 3G1 anjos@stanfordalumni.org
[2] Department of Electrical & Computer Engineering, University of Waterloo, Waterloo, Ontario, Canada N2L 3G1 vannelli@cheetah.vlsi.uwaterloo.ca

1 Introduction

The single-row layout problem (SRLP) 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. [12]. An instance of the SRLP consists of n one-dimensional facilities, denoted $1, \ldots, n$, with given positive lengths $\ell_1, \ldots, \ell_n$, 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 SRLP becomes a special case of the quadratic assignment problem, see e.g. [11]. Several applications of the SRLP 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. [10]. Furthermore, the SRLP is closely related to the linear ordering problem, which also has a number of practical applications, see e.g. [5, 6, 13].

The SRLP was first studied by Simmons [14] who proposed a branch-and-bound algorithm. Subsequently, Picard and Queyranne [12] developed a dynamic programming algorithm, and mixed integer linear programming models have also been proposed, most recently in [1]. While these algorithms

* Research partially supported by grant NAL/00636/G from the Nuffield Foundation (UK), grants 312125, 314668 and 15296 from the Natural Sciences and Engineering Research Council of Canada, and a Bell University Laboratories Research Grant.

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 SRLP have also been proposed. We refer the reader to the recent paper of Solimanpur et al. [15], and the references therein. However, these heuristic algorithms do not provide a guarantee of global optimality, or even some measure of the possible distance from optimality. Progress in obtaining such a measure was recently reported in [2], where non-trivial 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 SDP webpage [7] as well as the books [4, 16] 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.

The application of SDP to the SRLP was initiated in the aforementioned paper [2] which proposed an SDP relaxation as well as a heuristic which extracts a feasible solution to the SRLP from the optimal matrix solution to the SDP relaxation. Therefore, this SDP-based approach yields both a feasible solution to the given SRLP instance as well as a guarantee of how far it is from global optimality. The majority of the results reported in [2] were for fairly large instances, and were obtained using the spectral bundle solver SB [8, 9] which is able to handle very large SDPs, but with the drawback that it must run for several hours. The results reported in [2] showed that the SDP-based approach yields layouts that are consistently a few percentage points from global optimality for randomly generated instances with up to 80 facilities.

In this paper, we further study the computational performance of the SDP relaxation by considering smaller problems for which the SDP relaxation can be solved using an interior-point solver. This allows us to test a branch-and-bound (B&B) algorithm for the SRLP that solves the SDP relaxation from [2] at each node. Our results show that it is possible to compute solutions that are provably very close to global optimality (typically less than 1% gap) for randomly generated instances of the SRLP with up to 40 facilities with a reasonable amount of computational effort. More interestingly, the results also show that branching does not provide a significant improvement in the results obtained at the root node of the B&B tree, suggesting that although the SDP approach efficiently computes layouts that are provably very close to optimality, it would require a significant increase in computational effort to attain provable global optimality for this problem.

2 The Semidefinite Programming 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 $\frac{1}{2}\ell_i + D_\pi(i,j) + \frac{1}{2}\ell_j$, where $D_\pi(i,j)$ denotes the sum of the lengths of the facilities between i and j in the ordering defined by π. The problem is therefore to

$$\min_{\pi \in \Pi} \sum_{i<j} c_{ij} \left[\frac{1}{2}\ell_i + D_\pi(i,j) + \frac{1}{2}\ell_j \right]$$

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

Simmons [14] observed that if we rewrite the objective function as

$$\min_{\pi \in \Pi} \sum_{i<j} c_{ij} D_\pi(i,j) + \sum_{i<j} \frac{1}{2} c_{ij} (\ell_i + \ell_j)$$

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 possible to simplify the problem by considering only the permutations for which, say, facility 1 is on 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 SLRP proposed in [2] 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 $R_{ij} = -R_{ji}$. To accurately formulate the SRLP, 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 which can be formulated as a set of quadratic constraints:

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

This leads to the following formulation of the SLRP:

$$\begin{aligned} \min\ & K - \sum_{i<j} \tfrac{c_{ij}}{2} \left[\sum_{k<i} \ell_k R_{ki} R_{kj} - \sum_{i<k<j} \ell_k R_{ik} R_{kj} + \sum_{k>j} \ell_k R_{ik} R_{jk} \right] \\ \text{s.t.}\ & \\ & R_{ij} R_{jk} - R_{ij} R_{ik} - R_{ik} R_{jk} = -1 \text{ for all triples } i < j < k \\ & R_{ij}^2 = 1 \text{ for all } i < j \end{aligned}$$

where $K := \left(\sum_{i<j} \frac{c_{ij}}{2} \right) \left(\sum_{k=1}^{n} \ell_k \right)$. 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 SRLP.

To formulate the SLRP in the space of real symmetric matrices, let P denote the set of all pairs (i, j) such that $i < j$. Fixing an ordering of the elements of P, we can define the vector

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

where each p_k denotes a distinct element of P. Using v, we construct the rank-one matrix $X := vv^T$ whose rows and columns are indexed by P according to the ordering fixed above. By construction, $X_{p_i,p_j} = R_{p_i} R_{p_j}$ for all $p_i, p_j \in P$, and therefore we can formulate the SRLP as:

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

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. Removing the rank constraint yields the SDP relaxation. Note that in general the SDP problem only provides a lower bound on the optimal value of the SRLP, and not a feasible solution, unless the optimal matrix X^* happens to have rank equal to one. A rounding scheme specific to this problem was proposed in [2], and is discussed in the next Section.

3 Algorithm Description and Computational Results

The algorithm we report on here is a standard B&B algorithm that solves the SDP relaxation at every node. After solving each SDP, we apply the following

extension of the rounding procedure from [2]. If X^* is the optimal solution to the SDP relaxation, then each row of X^* corresponds to a specific pair of (i_1, j_1) of facilities. Therefore, for each row, 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

$$\omega_k = \frac{1}{2}\left(n + 1 + \sum_{j \neq k} R_{kj}\right)$$

for $k = 1, \ldots, n$, and hence obtain a permutation of $[n]$ by sorting these (in decreasing or increasing order, whichever satisfies $R_{i_1 j_1} = +1$). This approach improves on [2] by considering every row, rather than only the first row, thus obtaining a greater number of candidate layouts[3]. The best layout found so far is updated after each use of this heuristic.

After solving the relaxation and applying the rounding procedure at a node of the B&B tree, if the optimal solution X^* to the SDP problem is not rank-one, and the remaining subtree cannot be pruned, then the algorithm branches as follows. First, with $p_1 = (1, 2)$ being the pair corresponding to the first entry of v in the construction of the SDP, we set $R_{12} = +1$, and scan the first row of X^*, assigning value $X_{12,ij}$ to the variable R_{ij}, for every pair $(i, j) \neq (1, 2)$. Second, among all the variables R_{ij} not yet fixed and with absolute value less than 0.5 according to the assignment we just made, we choose the variable for which c_{ij} is largest. Any remaining ties are broken arbitrarily. This strategy is motivated by the fact that for the rank-one matrices, all the entries equal ± 1; hence entries with small magnitude are less desirable.

All the computational results were obtained on a 2.0GHz Dual Opteron with 16Gb of RAM, and the SDP problems were solved using the interior-point solver CSDP [3]. We ran the algorithm with a time limit of 3600 seconds, and with the possibility of exceeding this limit slightly if there was an SDP being solved when the time limit was reached, in which case the solver was allowed to run to completion. We solved 20 randomly generated instances of the SRLP for each of $n = 25, 30, 35, 40$, and the averages over the 20 instances for each value of n are reported in Table 1.

Our results suggest that using the algorithm described, the SDP-based approach is able to compute solutions that are provably very close to global optimality, typically with a gap of less than 1%, for randomly generated instances of the SRLP with up to 40 facilities. The results also show that branching does not provide a significant improvement in the results obtained at the root node of the B&B tree, suggesting that although the SDP approach efficiently computes layouts that are provably very close to optimality, it would require a significant increase in computational effort to attain provable global optimality for this problem. This last observation clearly illustrates the hardness of the SRLP.

[3] We thank Robert J. Vanderbei for suggesting this extension of the heuristic.

Table 1. Summary of computational results

Size of instances (n)	Time to solve root SDP (sec)	Gap to best layout found at root	Levels below the root completed within 1 hour	Total time to fully solve the completed levels	Gap to best layout found after total time
25	60.4	0.66%	3	2761.9	0.44%
30	260.3	1.11%	2	2498.7	0.97%
35	808.0	0.89%	1	3309.4	0.81%
40	2900.9	0.85%	0	2900.9	0.85%

References

1. A.R.S. Amaral. On the exact solution of a facility layout problem. *Eur. J. Oper. Res.*, to appear.
2. M.F. Anjos, A. Kennings, and A. Vannelli. A semidefinite optimization approach for the single-row layout problem with unequal dimensions. *Discr. Opt.*, 2(2):113–122, 2005.
3. B. Borchers. CSDP, a C library for semidefinite programming. *Optim. Methods Softw.*, 11/12(1-4):613–623, 1999.
4. E. de Klerk. *Aspects of Semidefinite Programming*, volume 65 of *Applied Optimization*. Kluwer Academic Publishers, Dordrecht, 2002.
5. M. Grötschel, M. Jünger, and G. Reinelt. A cutting plane algorithm for the linear ordering problem. *Oper. Res.*, 32(6):1195–1220, 1984.
6. M. Grötschel, M. Jünger, and G. Reinelt. Facets of the linear ordering polytope. *Math. Program.*, 33(1):43–60, 1985.
7. C. Helmberg. `http://www-user.tu-chemnitz.de/~helmberg/semidef.html`.
8. C. Helmberg and K.C. Kiwiel. A spectral bundle method with bounds. *Math. Program.*, 93(2, Ser. A):173–194, 2002.
9. C. Helmberg and F. Rendl. A spectral bundle method for semidefinite programming. *SIAM J. Optim.*, 10(3):673–696 (electronic), 2000.
10. S.S. Heragu and A. Kusiak. Machine layout problem in flexible manufacturing systems. *Oper. Res.*, 36(2):258–268, 1988.
11. W. Liu and A. Vannelli. Generating lower bounds for the linear arrangement problem. *Discrete Appl. Math.*, 59(2):137–151, 1995.
12. J.-C. Picard and M. Queyranne. On the one-dimensional space allocation problem. *Oper. Res.*, 29(2):371–391, 1981.
13. G. Reinelt. *The linear ordering problem: algorithms and applications*, volume 8 of *Research and Exposition in Mathematics*. Heldermann Verlag, Berlin, 1985.
14. D.M. Simmons. One-dimensional space allocation: An ordering algorithm. *Oper. Res.*, 17:812–826, 1969.
15. M. Solimanpur, P. Vrat, and R. Shankar. An ant algorithm for the single row layout problem in flexible manufacturing systems. *Comput. Oper. Res.*, 32(3):583–598, 2005.
16. H. Wolkowicz, R. Saigal, and L. Vandenberghe, editors. *Handbook of Semidefinite Programming*. Kluwer Academic Publishers, Boston, MA, 2000.