Intensification and Diversification with Elite Tabu Search Solutions for the Linear Ordering Problem

MANUEL LAGUNA
Graduate School of Business, University of Colorado, Boulder, CO 80309, USA
Manuel.Laguna@Colorado.Edu

RAFAEL MARTÍ
Departamento de Estadistica e Investigacion Operativa, Universidad de Valencia, Spain
Rafael.Marti@uv.es

VICENTE CAMPOS
Departamento de Estadistica e Investigacion Operativa, Universidad de Valencia, Spain
Vicente.Campos@uv.es

Latest revision: March 23, 1998.

ABSTRACT - In this paper, we develop a new heuristic procedure for the linear ordering problem (LOP). This NP-hard problem has a significant number of applications in practice. The LOP, for example, is equivalent to the so-called triangulation problem for input-output tables in economics. In this paper we concentrate on matrices that arise in the context of this real-world application. The proposed algorithm is based on the tabu search methodology and incorporates strategies for search intensification and diversification. For search intensification, we experiment with path relinking, a strategy proposed several years ago in connection with tabu search, which has been rarely used in actual implementations. Extensive computational experiments with input-output tables show that the proposed procedure outperforms the best heuristics reported in the literature. Furthermore, the experiments also show the merit of achieving a balance between intensification and diversification in the search.

1. Introduction

The practical significance of the linear ordering problem (LOP) has been well documented in the literature (see e.g. Grotschel, et al., 1984 and Chanas and Kobylanski, 1996). Solution methods for the LOP have been proposed since 1958, when Chenery and Watanabe outlined some ideas on how to obtain solutions for this problem. The interest on this problem has continued over the years, resulting in a book by Reinelt (1985) and in the most recent solution method due to Chanas and Kobylanski (1996).

Given a matrix of weights $E = \{e_{ij}\}_{m \times m}$, the LOP consists of finding a permutation p of the columns (and rows) in order to maximize the sum of the weights in the upper triangle. In mathematical terms, we seek to maximize:

$$C_E(p) = \sum_{i=1}^{m-1} \sum_{j=i+1}^{m} e_{p_i p_j} \ .$$

where p_i is the index of the column (and row) in position i in the permutation. Note that in the LOP, the permutation p provides the ordering of both the columns and the rows. The equivalent problem in graphs is that of finding, in a complete weighted graph, an acyclic tournament with a maximal sum of arc weights (Reinelt, 1985).

In economics, the LOP is equivalent to the so-called *triangulation problem for input-output tables*, which can be described as follows. The economy of a region (generally a country) is divided into m sectors and an $m \times m$ input-output table E is constructed where the entry e_{ij} denotes the amount of deliveries (in monetary value) from sector i to sector j in a given year. The triangulation problem then consists in permuting the rows and columns of E simultaneously such that the sum of the entries above the main diagonal is as large as possible. An optimal solution then orders the sectors in such a way that the suppliers (i.e., sectors that tend to produce materials for other industries) come first followed by the consumers (i.e., sectors that tend to be final-product industries that deliver their output mostly to end users).

Instances of input-output tables from sectors in the European and United States Economies can be found in the public-domain libraries LOLIB (1997) and Stanford Graph Base (Knuth, 1993), respectively.

2. Relevant Procedures

In this section, we describe two procedures for the LOP reported in the literature. We provide descriptions of these procedures because they will be used for comparison purposes in our computational experiments. Becker (1967) proposes a heuristic based on calculating quotients to rank each sector (using the interpretation from economics). In particular, for each sector i =1, ..., m the value

$$q_i = \frac{\sum_{k=1}^{m} e_{ik}}{\sum_{k=1}^{m} e_{ki}}$$

is calculated. The sector with the largest q-value is ranked highest. Then, the corresponding column and row are deleted from the matrix and the procedure is applied to the remaining sectors. A pseudo code of this method is shown in Figure 1.

Fig. 1 Becker's method.

```
Input:  E = {e_ij}_mxm
Output: p = (p_1, p_2, ..., p_m)

becker {
        I = {1, ..., m};
        j = 0;
        while (j < m) {
                k = argmax{q_i : i ∈ I};
                j = j + 1;
                p_j = k;
                I = I - {k};
        }
        return (p);
}
```

Becker's method is quite fast, and produces reasonable results considering its simplicity. We now describe the method developed by Chanas and Kobylanski (1996), which we will refer to as the CK procedure.

The CK method is based on the following symmetry property of the LOP. If the permutation $(p_1, p_2, \ldots, p_m)$ is an optimal solution to the maximization problem, then an optimal solution to the minimization problem is $(p_m, p_{m-1}, \ldots, p_1)$. In other words, when the sum of the elements above the main diagonal is maximized, the sum of the elements below the diagonal is minimized. The CK method utilizes this property to escape local optimality. In particular, once a local optimal solution p' is found, the process is re-started from the permutation $p'' = (p'_m, p'_{m-1}, \ldots, p'_1)$. The operation REVERSE is defined with this purpose, where $p'' = \text{REVERSE}(p')$.

In addition to REVERSE, the operations SORT and INSERT are utilized by the CK method. These operations are defined as follows.

$$\text{SORT}(p_1, p_2, \ldots, p_k) = \begin{cases} p_1 & \text{for } k = 1 \\ \text{INSERT}(p_k, \text{SORT}(p_1, \ldots, p_{k-1})) & \text{for } k > 1 \end{cases}$$

$$\text{INSERT}\left(i,\left(p_1,p_2,\ldots,p_{k-1}\right)\right)=\left(p_1,\ldots,p_{r-1},i,p_r,\ldots,p_{k-1}\right)$$

where $r \in \{1, 2, ..., k\}$ maximizes the value

$$\Delta C_E\left(i,r,\left(p_1,p_2,\ldots,p_{k-1}\right)\right)=\sum_{j=1}^{r-1} e_{p_j i}+\sum_{j=r}^{k-1} e_{i p_j} \ .$$

Figure 2 shows the pseudo code for the routine **bestsort**, which, through repeated calls to SORT, finds a local optimal ordering of the sectors.

Fig. 2 Bestsort routine.

```
Input:   E = {e_ij}_mxm and p
Output:  p¢

bestsort {
        p¢= p;
        do {
                p = p¢
                p¢= SORT(p);
        } while (C_E(p¢) > C_E(p));
        return (p¢);
}
```

The complete CK procedure is described by the pseudo code in Figure 3.

Fig. 3 CK procedure.

```
Input:   E = {e_ij}_mxm and p
Output:  p*

ck {
        p* = bestsort(p);
        do {
                p = p*;
                p* = bestsort(REVERSE(p));
        } while (C_E(p*) > C_E(p));
        return (p*);
}
```

Additional details and proofs related to the CK procedure can be found in Chanas and Kobylanski (1996). We now focus on the description of the tabu search procedure proposed in this paper.

3. Tabu Search Procedure

The tabu search (TS) technique is rapidly becoming the method of choice for designing solution procedures for hard combinatorial optimization problems. A comprehensive examination of this methodology can be found in the book by Glover and Laguna (1997). The following subsections describe three elements that are critical in the development of the tabu search procedure for the linear ordering problem (TS_LOP).

3.1 Insert Moves

Insertions are used as the primary mechanism to move from one solution to another in TS_LOP. We define INSERT_MOVE(p_j, i) to consist of deleting p_j from its current position j to be inserted in position i (i.e., between the current sectors p_{i-1} and p_i). This operation results in the ordering p', as follows:

$$p' = \begin{cases} (p_1, \ldots, p_{i-1}, p_j, p_i, \ldots, p_{j-1}, p_{j+1}, \ldots, p_m) & \text{for } i < j \\ (p_1, \ldots, p_{j-1}, p_{j+1}, \ldots, p_i, p_j, p_{i+1}, \ldots, p_m) & \text{for } i > j \end{cases}$$

Then the objective function value corresponding to p' can be obtained with the following calculation:

$$C_E(p') = \begin{cases} C_E(p) + \sum_{k=i}^{j-1} \left(e_{p_j p_k} - e_{p_k p_j} \right) & \text{for } i < j \\ C_E(p) + \sum_{k=j+1}^{i} \left(e_{p_k p_j} - e_{p_j p_k} \right) & \text{for } i > j \end{cases}$$

Note that the complexity for evaluating p' is in both cases $O(m)$. The following example illustrates the insertion mechanism. Suppose that the matrix corresponding to the permutation $p = (1,2,3,4,5,6,7)$ has the form shown below, and that the INSERT_MOVE(p_6, 2) must be evaluated.

$$E(p) = \begin{pmatrix} 0 & 12 & 5 & 3 & 1 & 8 & 3 \\ 6 & 0 & 3 & 6 & 4 & 4 & 2 \\ 8 & 5 & 0 & 5 & 7 & 0 & 3 \\ -2 & 7 & 2 & 0 & -3 & 6 & 0 \\ 8 & 0 & 3 & -1 & 0 & 4 & 1 \\ 9 & 1 & 6 & 2 & 13 & 0 & 4 \\ 2 & 9 & 4 & -5 & 8 & 1 & 0 \end{pmatrix}$$

The elements of the matrix that are inside of the rectangles are those needed to evaluate the objective function corresponding to the new permutation p' = (1, 6, 2, 3, 4, 5, 7). Since the objective function value for p is 78, the one for p' becomes:

$$C_E(p') = 78 + (1 - 4) + (6 - 0) + (2 - 6) + (13 - 4) = 78 + 8 = 86$$

It can also be verified that the best move involving p_6 is in fact INSERT_MOVE(p_6, 3), with a corresponding move value of 11. The move value is the difference between the objective function values after and before the move. In mathematical terms:

$$MoveValue = C_E(p') - C_E(p).$$

3.2 Neighborhood Definition

Two neighborhoods were considered during preliminary experimentation.

N_1 = {p′ : INSERT_MOVE(p_j, i), for j = 1, ..., m-1 and $i = j$+1}
N_2 = {p′ : INSERT_MOVE(p_j, i), for j = 1, ..., m and i = 1, 2, ..., j-1, j+1, ..., m}

N_1 consists of permutations that are reached by switching the positions of contiguous sectors p_j and p_{j+1}. N_2 consists of all permutations resulting from executing general insertion moves, as defined above. In conjunction with these neighborhoods, two strategies are defined. The *best* strategy selects the move with the largest move value among all the moves in the neighborhood. The *first* strategy, on the other hand, scans the list of sectors (in the order given by the current permutation) in search for the first sector (p_f) whose movement results in an strictly positive move value (i.e., a move such that $C_E(p') > C_E(p)$). The move selected by the *first* strategy is then INSERT_MOVE(p_f, i^*), where i^* is the position that maximizes $C_E(p')$. Note that for N_1, $i^* = f$+1, while for N_2, i^* is chosen from i = 1, 2, ..., f-1, f+1, ..., m. Therefore, the *first* strategy used in combination with N_1 is equivalent to searching for the first improving move in the neighborhood.

Combining the selection strategies with the neighborhood definitions results in four greedy local search procedures: *first*(N_1), *best*(N_1), *first*(N_2), and *best*(N_2). The results of preliminary experimentation with these procedures are reported in Table 1.

Table 1. Preliminary experimentation.

	***first*(N_1)**	***best*(N_1)**	***first*(N_2)**	***best*(N_2)**
Deviation	25.21%	24.16%	0.15%	0.19%
Num. of Opt.	0	0	11	11
CPU sec.	0,00	0,01	0,01	0,04

The data used to produce Table 1 consist of 49 instances from the problem library LOLIB. These instances correspond to real input-output matrices for which the optimal sector orderings are known. Table 1 reports the average deviation from optimality, the number of optimal solutions found and the computational effort corresponding to each of the greedy procedures. Note that

these procedures do not incorporate any tabu search elements, since the purpose of the experiment is to determine which neighborhood exploration to implement within TS_LOP. The greedy procedure *first*(N_2) is the most effective with *best*(N_2) a close second. This results seems to indicate that an effective search strategy results from searching for the best move associated with a given sector, as opposed to searching for the best move overall (as done by *best*(N_2)). This result can be explained by observing that *first*(N_2) tends to have a "slower" ascent to a local maximum, avoiding a premature entrapment in an inferior local maximum. This phenomenon has been observed in other applications (see e.g. Laguna, et. al 1994). Based on this finding, we partition N_2 into m N_2^j neighborhoods

$$N_2^j = \{p' : \text{INSERT\_MOVE}(p_j, i),\ i = 1, 2, ..., j\text{-}1, j\text{+}1, ..., m\}$$

associated with each sector p_j, for $j = 1, ..., m$. We therefore base our local search on choosing the best insertion associated with a given sector. (The rules on how to select a sector are outlined below.) We did not attempt to use general swaps as a move strategy, since the consecutive-sector swaps in N_1 did not perform better than general insertions.

3.3. Measure of Influence

For each sector, there are at most m-1 relevant elements (i.e., those elements that may contribute to the objective function value). The elements in the main diagonal are excluded because their sum does not depend on the ordering of the sectors. This indicates, that sectors should not be treated equally by a procedure that selects a sector for a local search (i.e., for search intensification). We define w_j as the weight of sector j as follows:

$$w_j = \sum_{i \neq j} \left(e_{ij} + e_{ji} \right).$$

Note that weight values do not depend on the permutation p, and therefore they can be calculated off-line (i.e., before the search begins). The weight values will be used to bias the selection of sectors during the tabu search intensification phase.

3.4 Basic Procedure

Starting from a randomly generated permutation p, the basic TS procedure alternates between an intensification and a diversification phase as described below.

Intensification Phase

An iteration in this phase begins by randomly selecting a sector. The probability of selecting sector j is proportional to its weight w_j. The move INSERT_MOVE(p_j, i) $\in$ N_2^j with the largest move value is selected. (Note that this rule may result in the selection of a non-improving move.) The move is executed even when the move value is not positive, resulting in a deterioration of the

current objective function value. The moved sector becomes tabu-active for *TabuTenure* iterations, and therefore it cannot be selected for insertions during this time.

The number of times that sector *j* has been chosen to be moved is accumulated in the value *freq*(*j*). This frequency information is used for diversification purposes. The intensification phase terminates after *MaxInt* consecutive iterations without improvement. Before abandoning this phase, the *first*(N_2) procedure is applied to the best solution found (during the current intensification). We denote this solution as $p^{\#}$, in contrast to p^* (the best solution found over the entire search). By applying this greedy procedure (without tabu restrictions), a local optimum is guaranteed as the output of the intensification phase.

Diversification Phase

This phase is performed for *MaxDiv* iterations. In each iteration, a sector is randomly selected, where the probability of selecting sector *j* is inversely proportional to the frequency count *freq*(*j*). The chosen sector is placed in the best position, as determined by the move values associated with the insert moves in N_2^j.

The procedure stops when *MaxGlo* global iterations are performed without improving $C_E(p^*)$. A global iteration is an application of the intensification phase followed by the application of the diversification phase.

3.5 Additional Intensification with Path Relinking

Path relinking has been proposed in the context of tabu search (see e.g. Glover and Laguna, 1997). However, path relinking has largely been ignored by practitioners and researchers alike. One of the few path-relinking implementations appears in Laguna and Martí (1997). In the current development, the path relinking strategy is implemented with the goal of strengthening the intensification phase.

The best solution found at the end of an intensification phase $p^{\#}$ (which not necessarily represents p^*, the best solution overall) is subjected to a relinking process. The process consists of making moves starting from $p^{\#}$ (the initiating solution) in the direction of a set of elite solutions (also referred to as guiding solutions). The set of elite solutions consists of the *EltSol* best solutions found during the entire search. The insertions used to move the initiating solution closer to the guiding solutions can be described as follows. For each sector p_j in the current solution:

1) Find the position *i* for which the absolute value of (*j*-*i*) is minimized, where *i* is the position that p_j occupies in at least one of the guiding solutions.

2) Perform INSERT_MOVE(p_j, *i*).

Suppose, for example, that the current solution to a 4-sector problem is given by (A, B, C, D) and that the solutions (B, C, A, D) and (B, A, D, C) are being used as the guiding solutions. Consider sector A in the current solution, where p_1 = A. The values of i to be considered are i = 3 (for the first guiding solution) and i = 2 (for the second guiding solution). Since the absolute value of (j-i) is minimized when i = 2, then the move INSERT_MOVE(p_1, 2) is executed. The current solution becomes (B, A, C, D) and the process continues.

During the path relinking phase, a number of intermediate solutions are generated, like the solution given by (B, A, C, D) in the example above. These intermediate solutions are good candidates for additional exploration by way of applying a local search procedure. We apply *first*(N_2) to intermediate solutions once every four path-relinking iterations. That is, the local search procedure is applied to one fourth of the intermediate solutions visited during path relinking. The rational for applying local search to every fourth solution is based on the fact that consecutive intermediate solutions differ only in the position of one sector. Therefore, a local search procedure applied to consecutive intermediate solutions would likely converge to the same local optimum. The path relinking process terminates when all the sectors have been considered.

3.6 Additional Long Term Diversification

A long term diversification phase is implemented to complement the diversification phase in the basic procedure. The long-term diversification is applied after *MaxLong* global iterations have elapsed without improving $C_E(p^*)$.

For each sector p_j, a rounded average position $\alpha(p_j)$ is calculated using the positions occupied by this sector in the set of elite solutions and the solutions visited during the last intensification phase. Then, m diversification steps are performed which insert each sector p_j in its complementary position m-$\alpha(p_j)$, i.e., INSERT_MOVE(p_j, m-$\alpha(p_j)$) is executed for j = 1, ..., m.

This strategy is inspired by the REVERSE operation developed by Chanas and Kobylanski (1996). We, however, incorporate information about solutions that have been recently visited (during the last intensification phase) and solutions of high quality that have been found during the search (elite solutions). Purposefully constructing solutions that are “far away” from those in the elite set constitutes a diversifying element that also complements the intensification goal of the path relinking strategy.

4. Computational Experiments

The path relinking and the long term diversification strategies were coded both separately and jointly with the purpose of assessing their relative merit. There are therefore four variants of the method:

TS	Basic procedure (intensification and diversification phases).
TS_PR:	TS and path relinking.
TS_LD:	TS and long term diversification.

TS_LOP: TS and path relinking and long term diversification.

The first experiment has the goal of finding appropriate values for the three critical search parameters: *TabuTenure*, *MaxInt*, and *MaxDiv*. For this purpose, we employ a full factorial design with 3 levels for each parameter, as given in Table 2.

Table 2. Design of experiment for search parameters.

Factor	Level 1	Level 2	Level 3
TabuTenure	$0.5\sqrt{m}$	$\sqrt{m}$	$2\sqrt{m}$
MaxInt	$0.5m$	m	$2m$
MaxDiv	$0.5m$	m	$2m$

We perform the experiment on the set of 49 instances in LOLIB, setting the rest of the search parameters to the following values:

MaxGlo = 100
MaxLong = 50
EltSol = 4

The 27 test resulted in the best setting of *TabuTenure* = $2\sqrt{m}$, *MaxInt* = *m*, and *MaxDiv* = 0.5*m*. The average percent deviation from optimality using this setting was 0.0007% (since 47 out of the 49 instances were solved optimally).

Next, we explored the effect of changing *EltSol* from its current value of 4 to 3 and 5. By definition (Glover and Laguna, 1997), the set of elite solution is small, so values larger than 5 are generally not recommended. The experiment revealed that the number of optimal solutions found in the LOLIB set drops to 42 and 44, when *EltSol* is set to 3 and 5 respectively. Therefore, we set *EltSol* = 4 in the rest of our computational testing.

With the search parameters set as indicated above, we proceed to compare the relative merit of our tabu search variants. We employ three sets of instances: (1) the 49 instances in LOLIB, (3) the 75 instances in the Stanford GraphBase (Knuth, 1993), and (3) 75 randomly generated instances. The set of Stanford GraphBase problems consists of 25 instances for each size of 40, 60 and 75. A uniform distribution with parameters (0, 25000) was used to generate the random instances of sizes 75, 150 and 200 (25 instances per size). Tables 3, 4 and 5 show, for each TS variant, the average objective function value, the average percent deviation from optimality, the number of optimal solutions, and the average CPU time (seconds on a Pentium 166 MHz). Since optimal solutions are not known either for the Stanford GraphBase problems or the random instances, the deviation in Tables 4 and 5 is reported considering the best solution found during the experiment. Also for these tables, the number of best solutions found is reported instead of the number of optimal solutions.

Table 3. Comparison of TS variants with LOLIB instances.				
	TS	**TS_PR**	**TS_LD**	**TS_LOP**
Obj. Function	22,040,159.4	22,040,160.9	22,041,257.7	22,041,261.5
Deviation	0.04%	0.04%	0.00%	0.00%
Num. of Opt.	30	30	44	47
CPU seconds	0.33	0.54	0.67	0.93

Table 4. Comparison of TS variants with Stanford GraphBase instances.				
	TS	**TS_PR**	**TS_LD**	**TS_LOP**
Obj. Function	6,032,093.76	6,032,546.88	6,033,122.75	6,033,124.09
Deviation	0.018%	0.010%	0.001%	0.001%
Num. of Best	35	40	59	66
CPU seconds	1.16	2.29	2.65	4.13

Table 5. Comparison of TS variants with random (0, 25000) instances.				
	TS	**TS_PR**	**TS_LD**	**TS_LOP**
Obj. Function	129,223,009	129,223,369	129,255,824	129,269,367.5
Deviation	0.065%	0.065%	0.038%	0.027%
Num. of Best	25	41	20	33
CPU seconds	10.79	17.94	13.07	20.19

Table 3 reveals that for the LOLIB set of problems, TS_PR is not significantly better than the simple TS procedure. However, the TS_LD variant improves the quality of the solutions found by TS and increases the number of optimal solutions found. The complete procedure TS_LOP is capable of finding 47 out of 49 optimal solutions with approximately 39% more time than TS_LD. The computational effort is reasonable for all variants, with the most demanding method not reaching 1 CPU second on the average.

Table 4 shows an improving trend in the number of optimal solutions found as we move from TS to TS_LOP. For these instances, TS_PR is superior to TS, however, the long-term diversification component seems to have a larger effect on solution quality. Once again, the full procedure TS_LOP yields the best results (at the expense of additional computational effort).

Table 5 gives some interesting results. While the percent deviation keeps improving as we move from TS to TS_LOP, the number of best solutions found does not follow the same pattern. In fact, TS_PR is able to contribute with 41 out of the 75 best known solutions. Some of the problems in this set seem to particularly benefit from a strategy that provides additional intensification during the search.

In the following set of experiments we compare the performance of TS_LOP with the methods by Chanas and Kobylanski (1996), Becker (1967) and a greedy procedure based on the *first*(N_2) local search. As before, we refer to Chanas and Kobylanski's method as CK, and as CK-10 to the application of the method from 10 randomly generated initial solutions. In a similar way, Greedy-10 refers to the application of the greedy method from 10 different starting solutions. We include in this experiment the Greedy-10 procedure for comparison purposes, because it gives us an

indication of the quality improvement achieved by employing a sophisticated search, such as the one embedded in TS_LOP, versus the approach of randomizing a very simple local search.

We compare these procedures using the same set of 199 instances of our previous experiments. Average behavior is reported in three separate tables (6 7, and 8), one for each set of problems. Average deviation refers to optimal solutions for the LOLIB set and best-known solutions for the Stanford GraphBase and the random (0, 25000) sets.

Table 6. LOLIB problems (49 instances).

	Greedy	Greedy-10	Becker	CK	CK-10	TS_LOP
Value	22,033,729.49	22,038,090.39	20,375,556.16	22,018,008,35	22,040,892.14	22,041,261.51
Deviation	0.15%	0.02%	8.95%	0.15%	0.02%	0.00%
No. of Optimal	11	22	0	11	27	47
CPU seconds	0.01	0.08	0.02	0.10	1.06	0.93

Table 7. Stanford GraphBase problems (75 instances).

	Greedy	Greedy-10	Becker	CK	CK-10	TS_LOP
Value	6,022,126.63	6,032,440.56	5,909,898.24	6,028,562.89	6,032,591.57	6,033,124.09
Deviation	0.18%	0.01%	2.04%	0.08%	0.01%	0.00%
No. of Best	3	20	0	4	22	70
CPU seconds	0.06	0.55	0.20	1.45	16.33	4.09

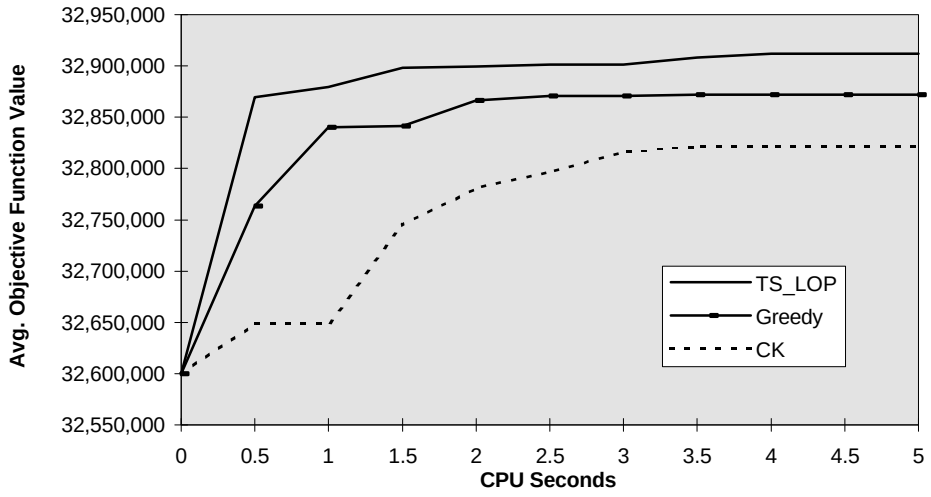
Table 8. Random (0, 25000) problems (75 instances).

	Greedy	Greedy-10	Becker	CK	CK-10	TS_LOP
Value	128,729,161.6	128,981,141	125,587,971.7	128,663,947.3	128,919,838	129,269,367.5
Deviation	0.47%	0.23%	3.08%	0.53%	0.28%	0.00%
No. of Best	0	2	0	0	0	73
CPU seconds	0.12	1.21	0.80	10.67	108.44	20.19

Becker's procedure is clearly inferior in terms of solution quality, although given its simplicity, its performance is quite acceptable. The performance of the Greedy and CK methods is very similar across the three problem sets. TS_LOP outperforms all other methods in terms of solution quality. This is most evident in the Random (0, 25000) set, where TS_LOP provides 73 of the 75 best-known solutions (compared to a maximum of 2 for all other methods). In terms of computational effort, TS_LOP remains superior to CK-10. Therefore, TS_LOP can be considered a superior solution method to the approach of applying the CK procedure from a number of randomly generated initial points.

To further study the behavior of the greedy procedure, the CK method and TS_LOP, we generated an additional set of 25 instances of size 75. The methods were run in a way that the best solution found was reported every 0.5 seconds. These data points were used to generate the performance graph in Figure 4. The superior performance of TS_LOP is once again made evident by Figure 4.

Fig. 4 Performance Graph.



The difference in quality between CK and the Greedy procedure may be due to the number of initial solutions used for each method. While the Greedy procedure can be applied 150 times during 5 seconds, the CK method can be applied only 10 times during the same amount of time.

5. Conclusions

In this paper we have developed an effective tabu search procedure for the linear ordering problem. The performance of the procedure has been assessed using 224 problem instances of several types and sizes. The procedure has been shown robust in terms of solution quality within a reasonable computational effort. The proposed method was compared with a recently developed procedure due to Chanas and Kobylanski (1996). The comparisons favor the proposed tabu search implementation.

An important goal of this research was to assess the merit of balancing diversification and intensification in a tabu search implementation. Our experiments show that long-term diversification by itself is an important component, since in all cases it enhanced the performance of the basic procedure. The additional intensification with path relinking, on the other hand, did not have (by itself) a major impact on solution quality in most cases. Our experiments also show that a balance between search diversification and intensification is achieved when both strategies are combined, resulting in an improved tabu search implementation.

Acknowledgments

The authors wish to thank Gerhard Reinelt for providing helpful clarifications on the implementation details of LOLIB and Fred Glover for his insightful comments on the path relinking strategy.

References

Becker, O. (1967) “Das Helmstädtersche Reihenfolgeproblem — die Effizienz verschiedener Näherungsverfahren” in:Computer uses in the Social Sciences, Berichteiner Working Conference, Wien, January 1967.

Chanas, S. and P. Kobylanski (1996) “A New Heuristic Algorithm Solving the Linear Ordering Problem,” *Computational Optimization and Applications*, Vol. 6, pp. 191-205.

Chenery, H. B. and T. Watanabe (1958) “International Comparisons of the Structure of Production” Econometrica, Vol. 26, p. 4.

Glover, F. and M. Laguna (1997) *Tabu Search*, Kluwer Academic Publisher.

Grotschel, M., M. Junger and G. Reinelt (1984), “A Cutting Plane Algorithm for the Linear Ordering Problem,” *Operations Research*, Vol. 32, No. 6, pp. 1195-1220.

Knuth, D. E. (1993) *The Stanford GraphBase: A Platform for Combinatorial Computing*, Addison Wesley, New York.

Laguna, M., T. Feo and H. Elrod (1994) “A Greedy Randomized Adaptive Search Procedure for the 2-Partition Problem,” *Operations Research*, vol. 42, no. 4, pp. 677-687.

Laguna, M. and R. Martí (1997) “GRASP and Path Relinking for 2-Layer Straight Line Crossing Minimization,” University of Colorado at Boulder.

LOLIB (1997) http://www.iwr.uni-heildelberg.de/iwr/comopt/soft/LOLIB/LOLIB.html.

Reinelt, G. (1985) *The Linear Ordering Problem: Algorithms and Applications*, Research and Exposition in Mathematics, Vol. 8, H. H. Hofmann and R. Wille (Eds.), Heldermann Verlag Berlin.

Intensification and Diversification with Elite Tabu Search Solutions for the Linear Ordering Problem

MANUEL LAGUNA
Graduate School of Business, University of Colorado, Boulder, CO 80309, USA
Manuel.Laguna@Colorado.Edu

RAFAEL MARTÍ
Departamento de Estadistica e Investigacion Operativa, Universidad de Valencia, Spain
Rafael.Marti@uv.es

VICENTE CAMPOS
Departamento de Estadistica e Investigacion Operativa, Universidad de Valencia, Spain
Vicente.Campos@uv.es

SCOPE AND PURPOSE — The linear ordering problem (LOP) has a wide range of applications in several fields. Perhaps, the best know application of the LOP occurs in the filed of economics. In this application, the economy (regional or national) is first subdivided into sectors. Then, an input/output matrix is created, in which the entry (*i*,*j*) represents the flow of money from sector *i* to sector *j*. Economists are often interested in ordering the sectors so that suppliers tend to come first followed by consumers. This is achieved by permuting the rows and columns of the matrix so that the sum of entries above the diagonal is maximized, which is the objective of the LOP.

In group decision making, for example, the linear ordering problem can be used to provide a ranking by paired comparison (or aggregation of individual preferences). A matrix entry (*i*,*j*) in this context may represent the strength of the preference that the group shows for option *i* over option *j*. Since the data may be inconsistent, there may not be a direct way of finding an ordering for the options. The solution to the corresponding LOP emerges as viable alternative for ranking the options under consideration.

Due to its combinatorial nature, the linear ordering problem has been shown to be hard (computationally speaking). While other computationally hard problems have capture the attention of researcher for many years (e.g., the traveling salesman problem), developing efficient solution procedures for the LOP has been somewhat neglected. The goal of our paper is two-fold: (1) to develop an efficient heuristic procedure for this problem, and (2) to experiment with the use of specialized strategies for search intensification and diversification, within the context of the search methodology that we have chosen to apply.