Variable Neighborhood Search for the Linear Ordering Problem

CARLOS G. GARCIA
Departamento de Economía de las Instituciones. Estadística Económica
Universidad de La Laguna, Spain. Cggarcia@ull.es

DIONISIO PÉREZ-BRITO
Departamento de Estadística, Investigación Operativa y Computación
Universidad de La Laguna, Spain. Dperez@ull.es

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

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

Latest revision: November 17, 2004.

Abstract

Given a matrix of weights, the Linear Ordering Problem (LOP) consists of finding a permutation of the columns and rows in order to maximize the sum of the weights in the upper triangle. This NP-complete problem can also be formulated in terms of graphs, as finding an acyclic tournament with a maximal sum of arc weights in a complete weighted graph. In this paper we first review the previous methods for the LOP and then propose a heuristic algorithm based on the Variable Neighborhood Search (VNS) methodology. The method combines two neighborhoods for an efficient exploration of the search space. We explore different search strategies and propose a hybrid method in which the VNS is coupled with a short term tabu search for improved outcomes. Our extensive experimentation with both real and random instances shows that the proposed procedure competes with the best-known algorithms in terms of solution quality, and has reasonable computing-time requirements.

KeyWords: Combinatorial Optimization, Metaheuristics, Linear Ordering Problem

[*] Corresponding author

1. Introduction

Given a matrix of weights $E = \{e_{ij}\}_{m\times m}$, the LOP consists of maximizing the expression:

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

where p_i is the index of the column (and row) in position i in the permutation. 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 of simultaneously permuting the rows and columns of E to make the sum of the entries above the main diagonal 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).

The motivation for our current development is to expand the VNS methodology (Hansen and Mladenovic, 1999) for global optimization by implementing different search strategies to apply its underlying framework to the linear ordering problem. As will be shown in the computational experiments, the proposed procedure provides high quality solutions within an extremely short computational time.

The next section summarizes the relevant literature on this problem. In Section 3 we provide a description of the different variants proposed to solve the LOP based on the VNS framework, including a hybrid method that combines the VNS with a short-term tabu search algorithm. These sections are followed by the results of our computational testing over a set of previously reported instances and the associated conclusions.

2. The Linear Ordering Problem

The linear ordering problem has generated a considerable amount of research interest since 1958, when Chenery and Watanabe outlined some ideas on how to obtain solutions for this problem. Table 1 summarizes the most relevant approaches to this problem.

Reference	**Contribution**
Chenery and Watanabe (1958)	Seminal paper
Becker (1967)	Greedy heuristic
Grotschel et al. (1984)	Exact Branch and Cut
Reinelt (1985)	Review
Chanas and Kobylanski (1996)	Multi-Start heuristic
Reinelt (1997)	LOLIB
Laguna et al. (1999)	Tabu Search
Mitchell and Borchers (2000)	Exact Cutting Plane
Campos et al. (2001)	Scatter Search

Table 1. Summary of relevant literature

Becker (1967) proposes a heuristic based on calculating quotients to rank each sector (using the interpretation from economics). The sector with the largest quotient is ranked highest. The corresponding column and row are then deleted from the matrix and the procedure is applied to the remaining sectors. Becker's method is quite fast and produces reasonable results considering its simplicity.

Grotschel et al. (1984) describe an exact solution algorithm for the Linear Ordering Problem that could be considered as the very first true branch and cut algorithm. This method exploits the linear description of the LOP polytope and is able to obtain optimal solutions for a set of real world instances. Reinelt (1985) summarizes the relevant methods and applications to this problem. This author also maintains a public-domain library, so-called LOLIB (1997), with 49 instances of input-output tables from sectors in the

European economy. This website also contains optimal solutions (obtained with the branch and cut mentioned above) for these instances. We will use these results in our computational experiments to measure the quality of the heuristic methods. There have been other important contributions in the context of exact methods for the LOP. For instance, Mitchell and Borchers (2000) propose a cutting plane algorithm based on a primal-dual interior point method to solve the first relaxation, and on the simplex method for the last few relaxations. Their computational results show the effectiveness of the proposed procedure. However, since our goal is to develop a heuristic method for this problem, we focus mainly on previous heuristic developments.

The multi-start method by Chanas and Kobylanski (1996) is based on an insertion mechanism that searches for the best position to insert a sector in the partial ordering under construction. Sectors are scanned according to the order of the current solution. When no further improvement is possible (hence a local optimal solution is found), the process is re-started from the reverse permutation of the local optimum found. The method is based on the symmetry property of the LOP, in which if the permutation (p_1, p_2, …p_m) is an optimal solution to the maximization problem, then an optimal solution to the minimization problem is given by the permutation (p_m, p_{m-1},…, p_1). It is expected that the re-starting from a reversed local optimum would induce a diversification component over the search.

Insertions are also used as the primary mechanism to move from one solution to another in the tabu search algorithm by Laguna et al. (1999). Specifically, INSERT_MOVE(p_j, i) consists of deleting sector 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:

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

The objective function value corresponding to p' is obtained with the following calculation:

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

Starting from a randomly generated permutation p, the short-term TS procedure alternates between an intensification and a diversification phase. At each iteration of the intensification phase, a sector is randomly selected. The probability of selecting sector j is proportional to an influence measure. The method scans the neighborhood $N(p_j)$ in search of the first move with a strictly positive value (i.e., a move such that $C_E(p') > C_E(p)$). The neighborhood $N(p_j)$ consists of all permutations resulting from the insertion of p_j in another position:

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

The move with the largest value is selected although it may result in the selection of a non-improving move (resulting in a deterioration of the current objective function value). The moved sector becomes tabu-active for a pre-established number of iterations, and therefore it cannot be selected for insertions during this time. In the diversification phase, the probability of selecting a sector is inversely proportional to the number of times that it has been previously selected. The basic procedure stops when a number global iterations (intensification phase followed by diversification phase) are performed without improving $C_E(p^*)$. We will refer to this short term memory tabu search method as ST_TS.

Laguna et al. (1999) also propose an extended tabu search method in which ST_TS is coupled with both long-term intensification and diversification. The intensification is based on the path relinking methodology, while the diversification is inspired on the symmetry property previously introduced by Chanas and Kobylanski (1996). The path relinking process consists of making moves starting from an initiating solution in the direction of a set of elite solutions also referred to as guiding solutions. Both the initiating solution and the set of elite solutions consist of the best solutions found during the ST_TS application. The long-term diversification constructs solutions that are “far away” from those in the elite set and constitutes a diversifying element that also complements the intensification goal of the path relinking strategy. We will refer to this complete tabu search algorithm as LT_TS. The authors compare both procedures, ST_TS and LT_TS, with the previous heuristic methods by Becker, and Chanas and Kobylanski in the 49 LOLIB instances as well as 75 randomly generated problems. The experiments show that Becker's procedure is clearly inferior and LT_TS outperforms all other methods in terms of solution quality.

Campos et al. (2001) adapt the Scatter Search evolutionary method to the LOP according to the so-called *template* given in Glover (1998), which is based on the following five elements:

1. Diversification Generator
   A constructive method based on modifying a measure of attractiveness proposed by Becker (1967) with a frequency measure that discourages sectors from occupying positions that they have frequently occupied in previous solution generations.
2. Improvement Method
   A local search "hill climbing" heuristic based on choosing the best insertion in $N(p_j)$ associated with a given sector j as described above.
3. Reference Set Update Method
   This is a generic (or context independent) Scatter Search element that builds and maintains the *reference set* consisting of the "best" solutions found (where the meaning of best includes not only quality but also diversity). Solutions gain membership to the reference set according to their quality or their diversity. See Laguna and Martí (2003) for a detailed description of this and the next element.
4. Subset Generation Method
   This is also a generic procedure and consists of creating different subsets of the reference set as a basis for implementing the subsequent combination method. Simple implementations apply the combination method only to pairs of solutions in the reference set, while more advanced SS designs consider subsets of different cardinalities for combination.
5. Solution Combination Method
   The method scans (from left to right) each reference permutation in the subset, and uses the rule that each reference permutation votes for its first element that is still not included in the combined permutation. The voting determines the next element to enter the first still unassigned position of the combined permutation.

The Diversification Generation Method is used to build a set P of 100 diverse solutions. The initial reference set is built according to the Reference Set Update Method. The reference set, *RefSet*, is a collection of both high quality solutions and diverse solutions that are used to generate new solutions by means of applying the Combination Method. The construction of the initial reference set starts with the selection of the 10 best solutions from P. These solutions are added to *RefSet* and deleted from P. For each solution in *P-RefSet*, the minimum of the distances to the solutions in *RefSet* is computed. The solution with the maximum of these minimum distances is then selected. This solution is added to *RefSet* and deleted from P and the minimum distances are updated. The process is repeated 10 times. The resulting reference set has 10 high quality solutions and 10 diverse solutions. The solutions in *RefSet* are ordered according to quality, where the best solution is the first one in the list. The subsets from the *RefSet* are created according to the Subset Generation Method and are selected one at a time in lexicographical order. The Solution Combination Method is applied to generate one trial solution from each subset. These trial solutions are subjected to the Improvement Method. The Reference Set Update Method is applied once again to build the new *RefSet* with the best solutions, according to the objective function value, from the current *RefSet* and the set of new improved solutions. If *RefSet* changes after the application of the reference set update method, then a new combination step is performed by applying the subset generation method to create subsets in which at least one solution is new. If the *RefSet* has not changed and no new solution qualifies to enter, the method finishes.

The authors compare the performance of their approach with the multi-start method developed by Chanas and Kobylanski and with the previously reported LT_TS tabu search. The experiments show that TS and the SS variants have very small average deviations from optimality for the LOLIB instances. Both procedures outperform previous approaches, including the multi-start method mentioned above.

Moreover, they are quite robust, as is evident from the negligible change in the deviation values across tables. We will consider both methods in our computational testing in Section 4.

3. The Variable Neighborhood Search Method

The Variable Neighborhood Search (VNS) is rapidly becoming a well-established method in metaheuristics (see for instance Hansen et al., 2001). VNS is based on a simple and effective idea: a systematic change of neighborhood within a local search algorithm. In this section we adapt the VNS to the LOP. We follow the description given in Hansen and Mladenovic (2003). To apply the method we first need to define different neighborhoods for our problem. Although VNS operates on any number k_{max} of neighborhoods, it has been observed that the best value for the parameter k_{max} is often 2. As is stated by Hansen and Mladenovic, VNS is based on three principles:

1. A local minimum with respect to one neighborhood is not necessarily so with another.
2. A global minimum is a local minimum with respect to all possible neighborhood structures.
3. For many problems local minima with respect to one or several neighborhoods are relatively close to each other.

Principle 2 is true for all the optimization problems. However, principles 1 and 3 may or may not hold depending on the problem at hand.

We have considered two classical neighborhoods in combinatorial optimization problems: switching and insertion. N_1 consists of permutations that are reached by switching the positions of contiguous sectors. N_2 consists of all permutations resulting from executing general insertion moves. Following the notation introduced in Laguna et al. (1999), given a sector p_j, the neighborhoods are:

$$N_1(p_j) = \{p' : \text{INSERT\_MOVE}(p_j, i), \text{ for } i = j\text{-}1, j\text{+}1\}$$
$$N_2(p_j) = \{p' : \text{INSERT\_MOVE}(p_j, i), \text{ for } i = 1, 2, ..., j\text{-}1, j\text{+}1, ..., m\}$$

Note that, as in other VNS implementations, both neighborhoods are nested ($N_1(p_j) \subseteq N_2(p_j)$). Note also that the tabu search method described in the previous section is based on $N_2(p_j)$.

Variable Neighborhood Descent

A first implementation to combine both neighborhoods in a deterministic way is given by the Variable Neighborhood Descent (VND). In its generic form, given two neighborhoods N_1 and N_2, the VND finds in each step the best neighbor p' of the current solution p ($p' \in N_k(p)$ where k is initially set to 1). If p' is better than p, then p is replaced with p' and k is set to 1; otherwise, k is set to 2. In the next steps, if k equals 1, the method proceeds in the same way and the best neighbor with respect to N_1 is selected to replace the current solution, if the new one improves it. If k equals 2 and the best solution p'found in $N_2(p)$ is better than the current solution p, then p is replaced with p' and k is set to 1; if p' does not improve p, the method finishes. In other words, the algorithm performs a local search for the best solution in N_1 and only resorts to performing one move in N_2 when the search is trapped in a local optima found in N_1. We will refer to this method as VND_best.

We have also tested a variant in which instead of finding the best solution in the neighborhood, the method scans it (in the order given by the current solution p) in search of the first solution p' that improves p. This variant will be denoted as VND_first.

Basic Variable Neighborhood Search

Basic VNS repeatedly performs three steps combining stochastic and deterministic strategies. In the first one, called *shaking*, a solution p' is randomly generated in $N_k(p)$. In the second one, a local search method is applied from p' to obtain a local optimum p''. In the third one, if p'' is better than p, p is replaced with p'' and k is set to 1; otherwise, k is switched (from 1 to 2 or from 2 to 1 in this case of two neighborhoods). As in the VND, k is initially set to 1 and the method resorts to N_2 when N_1 (now in combination with the local search) fails to improve on the current solution. However, if N_2 also fails to improve on the incumbent solution, instead of stopping the search, VNS sets k=1 and randomly selects another trial solution in N_1, repeating the three steps again. The sequence is repeated until a *Maxiter* number of consecutive iterations is performed with no further improvement.

As with the local search method in the VNS algorithm, we have implemented a greedy procedure based on neighborhood N_2, that in each iteration scans the list of sectors (in the order given by the current permutation) in search of the first sector (p_f) whose movement results in a strictly positive move value (i.e., the first improving move in the neighborhood such that $C_E(p') > C_E(p)$). The move selected by this *first* strategy is then INSERT_MOVE(p_f, i^*), where i^* is the position that maximizes $C_E(p')$. This local search was tested and compared with other alternatives in Laguna et al. (1999), showing remarkable results.

Extensions and Improvements

We are following a standard description of the Basic VNS (see for instance Hansen and Mladenovic, 2003) and we have experimentally found that, in this problem, the application of the local search from a randomly generated solution in the neighborhood of the incumbent solution, leads in many cases to a local optimum already visited. Specifically, for real instances we found that about 80% of the local optimum reached had already been visited in previous iterations, while in random instances this average is about 60%. We therefore propose two variants named K_VNS and Freq_VNS to extend the diversification given in the shaking step in order to escape from the basin of attraction of the incumbent solution, avoiding the cyclical tendency of the search.

The K_VNS method is implemented by modifying the shaking step in the basic VNS algorithm. Specifically, we randomly select a sector p_j in p and we move it to the best position, thus obtaining the best solution in $N_1(p_j)$. We repeat this movement k times before abandoning the shaking step. Since the objective in this step is to diversify the search, the moves are executed even when their values are not positive, resulting in a deterioration of the current objective function value. We then resort to the local search as in the basic VNS. This shaking step produces a solution that significantly moves away from the current solution p.

Diversification is the notion of expanding the search to unexplored regions in the solution space. This expansion consists of visiting solutions that have not been previously examined. Diversification strategies are generally based on either encouraging the incorporation of new elements or discouraging often examined elements. In particular, we use a frequency count $freq(i)$ in the Freq_VNS method to record the number of times sector i has been moved. Therefore, each time sector i is moved from one position to another in the shaking or the local search phase, we increment $freq(i)$ by one unit. We use this frequency count to generate a new solution in the shaking step. Since we want to diversify, we select the sectors j with a small frequency value $freq(j)$. Specifically, in the shaking step, we randomly select k sectors in the incumbent solution p to be moved, where k is a search parameter. The probabilistic selection rule is inversely proportional to the frequency count. The selected sectors are moved to the best available position (maximizing $C_E(p)$). As in the K_VNS, the move is executed even when the move value is not positive, resulting in a deterioration of the current objective function value, and after k moves the local search is applied from the solution obtained.

Hybrid Methods

The VNS method can be coupled with other procedures in many different ways to improve the performance of the “pure” algorithm. As in other VNS implementations, in this paper we target the hybridization that consists of replacing the local search with another procedure. We have considered two variants within this scheme.

In the VNS_VND algorithm, we replace the local search with the VND method. As will be shown in the next section, the Freq_VNS method outperforms the basic VNS, and therefore we consider the Freq_VNS as the base for the VNS_VND. In other words, the VNS_VND consists of three steps. In the first one, *shaking*, a new solution is generated by selecting k sectors in the incumbent solution according to their freq-values and moving them to the best available position. In the second step, *improvement*, we apply the VND method from the new solution. In the third step we check whether the solution generated by the VND method replaces the incumbent solution or not. The method repeats these three steps until a pre-specified limit is reached.

Finally, we consider a hybridization of the variable neighborhood search with the tabu search methodology. Following the scheme given above, in VNS_TS we replace the second phase, *improvement*, with the short term tabu search procedure ST_TS described in the previous section.

4. Computational Experiments

The procedures described in section 3 as well as the most relevant existing heuristics were implemented in C, and all experiments were performed on a Pentium IV personal computer at 2 GHz. The proposed variants and strategies were coded both separately and jointly with the purpose of assessing their relative merit. There are seven variants of the method:

VND_best	Variable Neighborhood Descent with the Best strategy.
VND_first	Variable Neighborhood Descent with the First strategy.
Basic VNS	Basic VNS directly adapted from Hansen and Mladenovic (2003)
K_VNS	Basic VNS in which shaking consists of *k* moves.
Freq VNS	Basic VNS in which shaking is performed according to frequencies.
VNS_VND	Hybrid VNS in which the local search is replaced with VND
VNS_TS	Hybrid VNS in which the local search is replaced with ST_TS

In our experiments we compare the performance of the VNS implementations for the linear ordering problem with three previous methods: the multi-start procedure by Chanas and Kobylanski (1996), CK, the tabu search procedure (Laguna et al. 1999), LT_TS, and the scatter search procedure (Campos et al. 2001), SS. As far as we know, these methods provide the best solutions known for this problem.

We have tested the procedures on four sets of previously reported instances:

(1) *LOLIB Instances*. These instances from the public-domain library consist of input-output tables from sectors in the European economy. Total number of instances is 49.

(2) *SGB Instances*. These instances from the Stanford GraphBase (Knuth, 1993) consist of input-output tables from sectors in the United States economy. The set has a total of 25 instances with 75 sectors.

(3) *Random Type I.* These instances are generated from a (0,100) uniform distribution. Reinelt (1985) introduced these instances. Campos et al. (2001) generated instances of sizes ranging from 35 to 200. There are 25 instances in each set for a total of 100. For the first set (size 35) the authors provide optimum solutions.

(4) *Random Type II.* These instances are generated by counting the number of times a sector appears in a higher position than another in a set of randomly generated permutations. For a problem of size *m*, *m*/2 permutations are generated. Chanas and Kobilansky (1996) introduced these instances. Campos et al. (2001) generated instances of sizes 100, 150 and 200. There are 25 instances in each set for a total of 75.

In our first preliminary experiment we compare simple local search methods. Specifically, we compare the VND with a greedy local search, LS, based on the N_2 neighborhood. Starting from a random solution, LS finds in each step the best neighbor *p*' of the current solution *p*. If *p*' is better than *p*, then *p* is replaced with *p*' and another step is performed; otherwise the method finishes. As was done with the VND method, we can consider two variants of the LS replacing the selection of the best with the first improvement in the neighborhood. Laguna et al. (1999) compared two local search methods, one based on neighborhood N_1 and the other on N_2, each one with the "first" and "best" variants, concluding that the local search based on N_2 is the most effective. Therefore we do not consider the local search based on N_1 in this experiment.

Combining the selection strategies with the method definitions results in four procedures: VND_first, VND_best, LS_first, and LS_best. The results of preliminary experimentation with these four procedures are reported in Tables 2 and 3. In this experiment we only consider those instances with known optimum. Table 2 shows the results on the LOLIB instances while Table 3 shows the results with the 25 random problems (type I). Both tables report the average deviation from optimality, the number of optimal solutions found and the computational effort corresponding to each of the greedy procedures.

Tables 2 and 3 show that LS and VND provide similar results. Considering the LOLIB instances, both present the same deviation with the best strategy, although VND_best is able to obtain 7 out of 49 optimal solutions while LS_best obtains 11. We also run both methods from 10 initial random solutions and VND_best obtains 24 optimal solutions while LS_best obtains 27. Table 3 shows that only VND_best is able to obtain one optimal solution out of the 25 random problems considered. However, LS_best presents the smallest average deviation from optimality with a value of 0.55. This experiment also confirms what is well known for this problem: the random instances are more difficult to solve than the real instances in which relationships between sectors are present. These instances are of a small size and with these simple methods it is difficult to observe running time differences; however, if we run them from different initial solutions it becomes clearer that VND saves time since it only resorts to N_2 when the search is trapped in N_1.

Table 2. Local search – 49 LOLIB instances.

	VND_first	**VND_best**	**LS_first**	**LS_best**
Deviation	0.20%	0.19%	0.29%	0.19%
Num. of Opt.	7	7	8	11
CPU sec.	0.001	0.000	0.001	0.001

Table 3. Local search – 25 Random Type I instances of size 35.

	VND_first	**VND_best**	**LS_first**	**LS_best**
Deviation	0.60%	0.63%	0.61%	0.55%
Num. of Opt.	0	1	0	0
CPU sec.	0.000	0.000	0.000	0.000

In our second experiment we compare the basic versions of the VNS method. In particular we consider the variants Basic VNS, K_VNS and Freq VNS described in Section 3. After preliminary experimentation, the parameter *k* in both latter methods has been set to 5 (this value provides the best solutions in terms of quality within low running times). Since these three procedures do not incorporate long term strategies, we compare them with the short-term tabu search method by Laguna et al. (1999) described in Section 2 (ST_TS). We have set the stopping parameter *Maxiter* in the VNS versions at 50 to approximate the running time consumed by the ST_TS method. As expected, if we increase this value, the performance of the methods improves considerably. Tables 4 and 5 show the results on the LOLIB and Random Type I instances with these four methods.

Table 4. Basic methods – 49 LOLIB instances.

	Basic VNS	**K_VNS**	**Freq VNS**	**ST_TS**
Deviation	0.02%	0.04%	0.05%	0.04%
Num. of Opt.	39	34	34	30
CPU sec.	0.008	0.007	0.007	0.009

Table 5. Basic methods – 25 Random Type I instances of size 35.

	Basic VNS	**K_VNS**	**Freq VNS**	**ST_TS**
Deviation	0.15%	0.07%	0.04%	0.05%
Num. of Opt.	10	17	17	14
CPU sec.	0.003	0.004	0.005	0.003

These tables show that the best solution quality is obtained by the VNS methods, which are able to match a larger number of optimal solutions than the short term TS considered. Specifically, in the LOLIB instances, Basic VNS matches 39 optimal solutions, K_VNS and Freq VNS 34, while ST_TS matches 30. On the other hand, on random Type I instances, Basic VNS matches 10 optimal solutions out of 25, K_VNS and Freq VNS 17, and ST_TS matches 14. However, ST_TS presents a similar average deviation from optima to the K_VNS and Freq VNS methods. All the methods are extremely fast considering that their running times are below 0.01 seconds.

In our next experiment we compare the hybrid VNS methods with the best procedures for the LOP. Tables 6 to 10 show, for each procedure, the average percentage deviation from optimality, the number of optimal solutions, and the average CPU time in seconds for each set of instances. Since optimal solutions are not known for the SGB and the large random instances, the deviation in Tables 8, 9 and 10 is reported considering the best solution found during each experiment. Also for these tables, the number of best solutions found is reported instead of the number of optimal solutions.

Table 6. Best methods – 49 LOLIB instances.

	VNS_TS	VNS_VND	LT_TS	SS	CK
Deviation	0.0082%	0.016%	0.0007%	0.0133%	0.02%
Num. of Opt.	46	39	47	42	27
CPU sec.	0.018	0.006	0.024	0.04	0.02

Table 7. Best methods – 25 Random Type I instances of size 35.

	VNS_TS	VNS_VND	LT_TS	SS	CK
Deviation	0.0221%	0.1212%	0.006%	0%	0.12%
Num. of Opt.	22	11	21	25	4
CPU sec.	0.012	0.005	0.014	0.023	0.004

Table 8. Best methods – 25 SGB instances of size 75.

	VNS_TS	VNS_VND	LT_TS	SS	CK
Deviation	0.0104%	0.0135%	0.0018%	0.0023%	0.0113%
Num. of Best	14	9	14	15	0
CPU sec.	0.052	0.031	0.090	0.153	0.64

Table 9. Best methods – 75 random type I instances (*m* = 100, 150 and 200).

	VNS_TS	VNS_VND	LT_TS	SS	CK
Deviation	0.1574%	0.2129%	0.0590%	0.0105%	0.277%
Num. of Best	8	3	10	54	0
CPU sec.	0.305	0.330	0.417	0.709	1.34

Table 10. Best methods – 75 random type II instances (*m* = 100, 150 and 200).

	VNS_TS	VNS_VND	LT_TS	SS	CK
Deviation	0.0016%	0.0060%	0.0012%	0.0015%	0,0096%
Num. of Best.	39	9	34	22	0
CPU sec.	0.220	0.377	0.269	0.457	0.88

The results of our VNS variants are obtained with computational efforts that average less than 0.4 seconds. The performance of the CK method is clearly inferior with average deviations several orders of magnitude larger than those achieved by the other approaches. However, it is a simple heuristic and its results are quite acceptable considering its simplicity. Considering the metaheuristics, they do not perform in a similar way across instances types. Specifically, table 6 shows that in the LOLIB problems the tabu search algorithm, LT_TS, is able to obtain 47 optimal solutions out of 49 instances in 0.024 seconds while the VNS variants, VNS_TS and VNS_VND, obtain 46 and 39 in 0.018 and 0.006 seconds respectively. The performance of the SS method in this experiment is clearly inferior in terms of quality considering its running time. Table 7 shows that the best solution quality of the 25 Random Type I small instances is obtained with the SS method, which is able to match all optimal solutions. However, it employs significantly longer running times than the other approaches. VNS_TS is very competitive, considering its 22 optimal solutions achieved in 0.012 seconds (which compares favorably with the 21 optima of the LT_TS method achieved in 0.014 seconds).

Tables 8, 9 and 10 show the results for large instances in which the optimal solution is not known. As in the previous experiments, SS obtains very good solutions (especially in random type I instances) but it employs longer running times than the other methods. VNS_TS and LT_TS are clearly the best methods in terms of solution quality achieved within small running times. Both obtain the same number of best solutions in the SGB instances, although LT_TS presents a smaller average percent deviation and a larger computational time than VNS_TS. On the other hand, VNS_TS obtains 8 best solutions and 0.15% average percent deviation in the random type I instances, while LT_TS obtains 10 best solutions and 0.06% average percent deviation. Results in random type II instances are different since VNS_TS is able to obtain 39 best solutions in 0.22 seconds of running time, which compares favorably with all the other methods considered.

Running times in these experiments have been set to match previous reported experiments (see Laguna et al. 1999). However, if we allow the methods to run for longer times, better solutions are obtained. Specifically, we have run the LT_TS and VNS_TS methods with the stopping parameter set to 10,000 (instead of the 100 considered above). In the 49 LOLIB and 25 SGB instances with optimum known, both methods are able to match all optimal solutions within less than 1 second of computer time. The tables in the appendix show the results (best value and run time in seconds) for the rest of the instances (which are available at http://www.uv.es/~rmarti).

Conclusions

The objective of our study has been to expand and advance the knowledge associated with the implementation of variable neighborhood search procedures. Unlike other local search based methods, such as tabu search, this methodology has not yet been extensively studied. In particular, we have undertaken to examine the adaptation of VNS to solve a well known hard optimization problem: the linear ordering problem. We have tested different variants of the procedure as well as some hybrid methods.

Overall experiments with 249 instances were performed to assess the merit of the procedures developed here. The results of our computational experiments reveal that the strategies implemented within a relatively simple variable neighborhood procedure are capable of producing good solutions. Moreover, we have explored some mechanisms to overcome the limitation of the basic design, as well as two different ways to hybridize the method with other metaheuristics to obtain high quality solutions. In particular, the combination of the VNS with a short-term memory tabu search has been shown to be robust in terms of solution quality within a reasonable computational effort. This method was compared with a recently developed tabu search and a scatter search procedure. The comparison favours the proposed VNS implementation.

Acknowledgments

Research by Vicente Campos and Rafael Martí is partially supported by the *Ministerio de Educación y Ciencia* (ref. TIC2003-C05-01) and by the *Agencia Valenciana de Ciencia y Tecnologia* (ref. GRUPOS03/189).

References

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

Campos, V., F. Glover, M. Laguna and R. Martí (2001), An Experimental Evaluation of a Scatter Search for the Linear Ordering Problem, *Journal of Global Optimization* 21, pp. 397-414

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. (1998) “A Template for Scatter Search and Path Relinking,” in *Artificial Evolution, Lecture Notes in Computer Science* 1363, J.-K. Hao, E. Lutton, E. Ronald , M. Schoenauer and D. Snyers (Eds.), Springer, pp. 13-54.

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.

Hansen, P. and Mladenovic, N. (1999), "An Introduction to variable neighborhood search", in Metaheuristics: Advances and trends in local search paradigms for optimization, Voss, S., Martello, S., Osman, I.H. and Roucairol, C. (Eds.), pp. 433-458.

Hansen, P. and Mladenovic, N. (2003), "Variable neighborhood search", in Handbook of Metaheuristics, Glover, F. and Kochenberger, G. (Eds.), pp. 145-184.

Hansen, P., Mladenovic, N. and Pérez-Brito, D. (2001), "Variable Neighborhood Decomposition Search" Journal of Heuristics vol. 7, no. 4, pp. 335-350.

Laguna, M., Martí, R., 2003. Scatter Search. Methodology and Implementations in C. Kluwer Academic Publishers.

Laguna, M., R. Martí and V. Campos (1999) "Intensification and Diversification with Elite Tabu Search Solutions for the Linear Ordering Problem," *Computers and Operations Research* 26, pp. 1217-1230.

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

Mitchell, J.E. and B. Borchers (2000), "Solving linear ordering problems with a combined interior point/simplex cutting plane algorithm", In High Performance Optimization, H. Frenk et al. (Eds.), pp. 349-366, Kluwer Academic Publishers.

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.

Appendix

SGB (size 75)	LT_TS (value)	LT_TS (CPU sec.)	VNS_TS (value)	VNS_TS (CPU sec.)
1	6144679	5.22	6144679	4.84
2	6100491	5.19	6100491	4.72
3	6165775	4.03	6165775	4.63
4	6154958	6.23	6154958	4.89
5	6141070	4.11	6141070	4.24
6	6144055	5.61	6144055	4.20
7	6142899	4.05	6142899	4.31
8	6154094	7.02	6154094	5.63
9	6135459	3.84	6135459	4.38
10	6149271	4.13	6149271	4.92
11	6151750	4.23	6151750	4.53
12	6150469	4.56	6150469	5.08
13	6156935	7.64	6156935	4.80
14	6149693	5.69	6149693	4.86
15	6150331	7.34	6150331	6.45
16	6164959	4.48	6164959	4.89
17	6163483	5.95	6163483	5.77
18	6063548	4.20	6063548	5.00
19	6150967	4.41	6150967	4.67
20	6152224	4.14	6152224	4.53
21	6159081	4.16	6159081	4.34
22	6127019	4.25	6127019	4.78
23	6136362	3.81	6136362	4.66
24	6168513	5.31	6168513	4.59
25	6150026	3.80	6150026	4.33

Random type I

	Size 100				Size 150				Size 200			
	LT_TS		VNS_TS		LT_TS		VNS_TS		LT_TS		VNS_TS	
1	271622	6.25	271549	5.56	603998	15.53	603406	11.77	1066545	34.84	1065079	20.66
2	271170	7.23	271106	5.83	605999	14.42	606406	11.78	1067416	37.14	1068188	59.63
3	273824	6.41	272794	6.38	605225	19.91	604773	23.20	1066618	61.27	1065538	21.09
4	271160	8.89	270978	5.56	603964	19.89	603436	12.44	1068817	26.91	1067354	19.64
5	272946	11.61	272459	7.53	603634	15.36	602399	14.08	1067804	33.25	1066589	21.52
6	270217	6.66	270326	5.52	602881	23.11	602138	11.84	1066104	30.61	1063958	54.75
7	273785	10.92	272892	5.67	606175	16.36	605758	12.80	1068049	47.42	1066199	34.91
8	273452	7.41	272637	7.41	612316	18.17	611411	33.58	1070932	36.41	1069708	26.67
9	273480	6.44	273326	8.17	607992	24.02	607846	12.97	1068787	29.91	1067518	25.77
10	273066	6.52	273066	5.91	608651	17.91	607272	18.03	1070623	30.28	1068891	25.81
11	270882	6.20	270671	5.72	602967	17.19	601710	12.53	1066238	44.67	1063600	24.53
12	270916	9.33	270698	11.25	605220	19.69	603997	13.91	1069983	43.64	1067668	20.59
13	271804	6.19	271695	5.70	605124	16.89	604009	18.52	1064734	32.25	1063454	22.83
14	269376	6.52	269048	9.19	605464	21.63	603980	12.06	1068576	32.67	1066317	44.80
15	274847	6.22	274403	5.67	608996	20.22	607691	12.78	1071280	28.72	1069335	22.36
16	273216	6.64	273207	6.92	606339	18.30	605393	15.03	1069493	35.09	1067831	37.02
17	273025	6.78	272735	6.06	605411	24.61	604338	11.66	1069387	58.95	1065097	23.06
18	270951	6.50	270892	5.92	603312	26.52	602122	12.06	1068233	37.34	1065858	57.52
19	270650	7.45	270648	5.81	602956	16.58	601845	12.11	1065566	30.83	1065582	26.11
20	274625	6.78	274111	11.24	605873	18.38	605116	11.69	1068789	48.53	1068946	56.81
21	274582	7.14	274197	9.25	606705	22.84	606649	35.36	1073835	31.52	1074094	25.44
22	272059	7.33	272026	5.78	604914	18.84	604541	16.95	1064220	47.42	1062753	38.88
23	271970	8.19	271881	7.19	605898	17.45	605432	12.16	1067725	40.19	1067148	38.30
24	271912	7.36	271809	6.19	606704	22.13	605205	11.25	1069591	35.33	1068543	24.63
25	270764	8.00	270477	6.55	605900	16.52	605737	11.72	1067428	29.25	1067090	26.13

Random type II

	Size 100				**Size 150**				**Size 200**			
	LT_TS		**VNS_TS**		**LT_TS**		**VNS_TS**		**LT_TS**		**VNS_TS**	
1	135648	4.70	135648	5.33	454420	13.02	454420	14.83	1063170	21.36	1063154	25.22
2	137192	4.89	137192	5.47	453199	13.19	453194	13.27	1061259	21.08	1061259	38.59
3	135865	4.75	135865	5.39	451061	12.41	451062	14.75	1059687	21.92	1059657	20.97
4	135962	4.86	135962	5.50	453473	13.30	453472	16.98	1064725	23.78	1064711	30.64
5	135384	4.73	135384	5.42	456476	11.81	456476	24.45	1064500	24.23	1064482	20.74
6	135505	5.08	135505	7.91	454210	11.91	454204	16.97	1059401	27.81	1059383	35.86
7	136468	4.75	136468	5.42	453249	13.50	453248	13.20	1064243	23.81	1064271	28.14
8	134686	5.56	134686	5.50	449718	12.47	449717	13.48	1064468	23.17	1064484	23.61
9	136759	5.50	136751	5.44	451618	12.58	451618	17.03	1059821	22.59	1059819	20.30
10	136225	4.86	136225	6.03	450615	12.80	450617	17.06	1064348	20.19	1064356	24.63
11	135296	4.80	135296	5.69	452677	12.28	452672	15.34	1063003	21.52	1063003	20.53
12	136262	4.80	136262	5.36	452277	14.31	452271	14.47	1065731	27.13	1065733	25.84
13	136840	4.84	136840	5.44	453659	12.45	453655	16.13	1057456	26.94	1057470	19.55
14	135722	5.00	135722	5.27	450212	12.47	450212	14.94	1061322	21.53	1061300	25.16
15	134902	4.70	134898	5.95	454950	12.67	454941	20.50	1059016	23.56	1059010	45.08
16	137001	5.22	137001	13.08	452369	12.83	452369	13.31	1062815	21.28	1062795	25.09
17	136284	4.92	136284	5.83	451149	13.02	451149	19.77	1054501	21.66	1054495	34.38
18	136386	4.89	136386	7.66	452872	12.78	452872	16.58	1065141	25.64	1065131	39.25
19	137389	5.06	137389	6.81	454457	13.28	454448	12.08	1057399	28.39	1057347	20.61
20	135435	5.14	135433	6.64	449790	15.73	449786	13.74	1062760	22.19	1062758	20.59
21	135024	5.52	135016	5.47	451283	15.11	451285	13.23	1060805	26.69	1060765	20.17
22	136592	4.84	136592	5.91	450816	12.11	450816	11.77	1062479	36.50	1062487	37.16
23	135632	4.84	135632	5.70	453797	13.17	453795	12.06	1061401	23.61	1061421	34.16
24	135195	4.95	135195	5.58	451160	13.58	451160	12.73	1065109	21.41	1065093	20.44
25	134612	4.88	134612	5.41	453286	15.02	453281	12.25	1063929	21.27	1063929	24.63