*Belgian Journal of Operations Research, Statistics and Computer Science Vol. 34 (3) 94*

# Using metaheuristics in solving dispatching problems

J.-M. Godart

Université de Mons–Hainaut
Faculté des Sciences économiques et sociales
Place du Parc 20
7000 Mons, Belgium

**Abstract**

This paper presents some aspects of delivery routes organization in companies. It also explains why metaheuristics can be very useful for dispatching computerization. Two main metaheuristics are presented : Tabu Search and Simulated Annealing. We briefly present the way simple metaheuristics can be adapted to "solve" real problems, which are remote from standardized ones such as T.S.P. The approach is illustrated by its application to a case with time windows constraints.

**Keywords :** Delivery routes, dispatching, metaheuristics, modules, simulated annealing, tabu search, T.S.P., m.T.S.P.T.W.

## 1. Introduction [1]

There are many algorithms aimed at solving standardized problems (T.S.P., V.R.P.,...) related to the organization of delivery routes (dispatching). More precisely, the purpose of these methods is to map out a route for the vehicles (and principally road vehicles) that go and deliver the goods of the company to the different shops or clients : which vehicle should we send ? at what time ? to which places ? in which order ? from which warehouse ? ...

These algorithms range from very simple ones (based more on a philosophy, a logical process, than on mathematics) to very elaborate ones (based on the most advanced mathematical developments). Yet, they can only rarely be applied in practice.

If one of these algorithms matched exactly the practical case which has to be solved, our advice would be to use the most efficient one, and therefore the most elaborate one, as far as mathematics is concerned.

On the contrary, and it's the most common case, if the situation is peculiar enough for no existing algorithm being able to solve it, we'd find it better to forget mathematical specificities of the algorithms and to base ourself only on the underlying philosophy of these algorithms.

The existing algorithms, even if they are very elaborate, are then only "guides" for solving most problems that occur in practice.

Moreover, in the three Belgian companies of which we studied the dispatching organization - Delhaize, Fina and Texaco - no routing software (that is a software used to plan delivery routes, on the basis of constraints, objectives and data of the company) was used ; Delhaize only had decided to buy such a software.

Therefore, we must conclude, without making generalizations, that these three companies consider that existing theoretical solutions are partly or totally incompatible with reality and that they prefer relying on logic, memory, willingness and experience of their dispatchers.

The relative incompatibility between theory and practice is certainly the main reason for which dispatching algorithms are little used. Another reason can be found in the fact that these algorithms are awfully time-consuming and it has been only a few years since computers - which are here the essential link between theory and practice - can offer a computing speed which is sufficient for solving large problems (with many constraints, delivery points,...).

Nevertheless, the incompatibility between practice and theory - and then the software using it - is not absolute ; it comes partly from a wrong way of considering the dispatching computerization.

Indeed, the output of routing softwares is often emphasized as being a final result instead of a "gross" result which has to be improved on later, according to several factors (e.g. human factors which can hardly be quantified).

---

1 Based on (15), (16) and (23).

When organizing the dispatching in the usual way, that is manually, dispatchers try first to create routes by grouping delivery points (shops, clients,...), according to their location (while already trying to satisfy most of the constraints) ; the routes which have been obtained are then modified in an attempt to satisfy all the constraints, without disturbing too much what has just been done.

If routing softwares can't replace dispatchers - and will probably never be able to do so as long as artificial intelligence won't be a reality -, they can make their task easier by making "mistakes" in their place. Indeed, if we split the task of dispatchers in two steps, firstly the creation of gross routes, with some violated constraints, and secondly the improvement of these routes, then dispatching softwares may simplify and even eliminate the first stage. On this point, routing softwares can thus be of real interest.

Moreover, when we know that in some oil companies, for example, the average global cost of transportation with one's own fleet is about 70 Bef/km, it's obvious that software development is very important, if these softwares could make every route a few kilometers shorter ; so it would be possible to cut down on time, energy and money needed for delivering the products.

If using routing software doesn't make it possible to spare money by reducing the number of kilometers, it enables the company to take more systematically into account some of its constraints or objectives. Therefore the company will benefit from it, for example in terms of customers' satisfaction. This will also make the company more competitive.

Finally, our transportation study of Delhaize reveals that some companies could have profit in routing softwares for other specific purposes (for "control" purposes, as far as Delhaize is concerned).

As we can see, there is many a reason why research in routing algorithms is so active.

## 2. Metaheuristics [2]

It is clear from what has just been said that algorithms aiming at "solving" dispatching problems can be of some interest, provided these algorithms are flexible enough and are considered only as tools.

Besides usual algorithms - exact or heuristic ones -, metaheuristic solutions are getting more and more used. These are more "solving philosophies" than genuine "solving methods" (algorithms). They have in common with heuristics the fact that the optimality of the solution obtained is sacrificed for the benefit of computing time ; yet, they are different from usual heuristic methods as they can apply to a large range of problems, simply by adapting some parameters (called here "technical choices").

With metaheuristics it becomes possible to "build" algorithms which are as close as possible to the practical situations we have to modelize ; as a result, algorithms can easily be modified to meet new requirements which could appear in practice.

---

2 Based on (20) and (21).

Among existing metaheuristics, we have chosen two of them, which are probably the most commonly used : "tabu search" and "simulated annealing". Before presenting these metaheuristics, we'll remind the reader - very briefly - what the "steepest descent strategy" is, on which these two metaheuristics are based.

## 3. Steepest descent strategy [3]

When we want to find $\alpha^* \in S$ such that :

$$f(\alpha^*) = \min_{\alpha \in S} f(\alpha)$$

where S is a finite set, with at least one item ($S \neq \varnothing$)
f is an application from S into $\mathbf{R}$

one of the most simple methods we could think of is probably the "steepest descent strategy". This method consists in applying explicit enumeration, on an iterative way, on subsets of S. At each iteration of the "descent strategy", the subset of S that we consider for the explicit enumeration, is in fact made of items of S close to that obtained as "optimum" for the previous iteration.

So, if at the $i^{th}$ iteration, explicit enumeration determines item $\alpha_i^*$ as optimum, the $(i + 1)^{th}$ iteration will consist in finding $\alpha_{i+1}^*$ such that :

$$f(\alpha_{i+1}^*) = \min_{\alpha \in V(\alpha_i^*)} f(\alpha)$$

where $V(\alpha_i^*)$ is the subset mentioned, made of items of S close to $\alpha_i^*$.

$V(\alpha_i^*)$ is thus what we call a "neighbourhood" of $\alpha_i^*$.

This method is a local search strategy consisting in exploring a neighbourhood whose structure has been defined by a function V :

$$V : \alpha \rightarrow V(\alpha) \qquad \text{where } \alpha \in S \text{ and } V(\alpha) \subset S$$

We'll assume that $\alpha \notin V(\alpha)$.

---

3 Based on (20), (21) and (24).

The steepest descent strategy has a natural end : the search stops when the best item found in a given neighbourhood is not "better" than the item of which we are considering the neighbourhood.

Of course, the algorithm as it has just been presented is only the philosophy of the method ; in order to use it in a specific case (such as T.S.P.), several technical choices have to be made. These choices may influence the efficiency of the method considerably and therefore much attention should be devoted to them.

Whatever the combinatorial optimization (minimization) problem it is applied to, the steepest descent strategy presents an important failure : it can't escape from local minima.

Indeed, let's assume that at the end of an iteration we have obtained $\alpha^*$ such that :

$$f(\alpha^*) \leq f(\alpha) \qquad \forall\, \alpha \in V(\alpha^*)$$

but that $\alpha^*$ is not the global minimum, that is :

$$\exists\, \alpha^{**} \in S \text{ such that } f(\alpha^{**}) < f(\alpha^*)$$

The steepest descent strategy can't escape from $\alpha^*$, which is then the final result of the algorithm.

The steepest descent strategy thus belongs to the category of methods progressing monotonously from an admissible solution ; the common feature of these methods is indeed that they can be blocked by meeting a local minimum.

In order to improve the steepest descent strategy, we should see to it that it could escape from an item $\alpha^*$ different from the global minimum but such that :

$$f(\alpha^*) \leq f(\alpha) \qquad \forall\, \alpha \in V(\alpha^*)$$

That's here that both metaheuristics "tabu search" and "simulated annealing" come into play by making it possible, each on its own way, to elude such a situation.

## 4. Tabu search [4]

The "tabu search" method, initially devised in 1977, was developed at the same time by F. Glover (1986), P. Hansen and B. Jaumard. Relatively close to the fundamental principles of artificial intelligence (training theory,...), this method memorizes the way followed for the search, to avoid considering solutions which would already have been evaluated.

If we simplify as much as possible, the "tabu" metaheuristic is a kind of improved steepest descent strategy. By means of "tabu search" the search can go beyond the first local minimum encountered.

---

[4] Based on (3), (20), (21), (22), (27) and (28).

The principle of "tabu search" is simple :

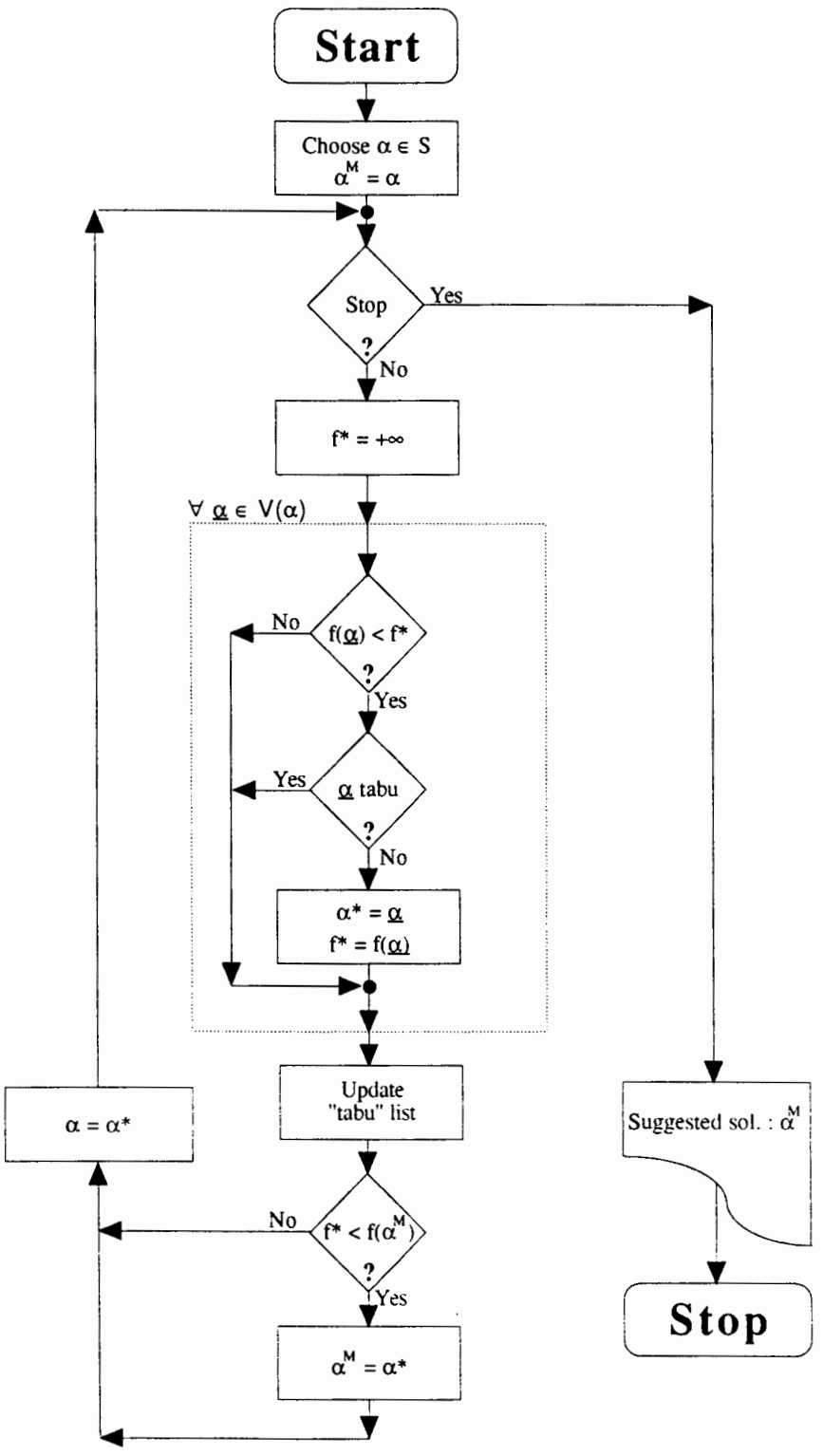
> The steepest descent strategy still selects, at iteration #i, the "best" item $\alpha_i^*$ from $V(\alpha_i)$ - where $\alpha_i = \alpha_{i-1}^*$ -, which will be used as $\alpha_{i+1}$ at the next iteration. The search stops when at iteration #j the best item $\alpha_j^*$ of $V(\alpha_j)$ is not "better" than $\alpha_j$.
>
> "Tabu search" will follow exactly the same process but will not stop at iteration #j (where the best item $\alpha_j^*$ found in $V(\alpha_j)$ is not "better" than $\alpha_j$). In such a situation, "tabu search" will choose $\alpha_j^*$ - in spite of its being worse than the item considered before - in order to use it as $\alpha_{j+1}$ at the next iteration. When meeting a local minimum, "tabu search" won't thus be blocked as it will make it possible to select "worse" solutions (that is solutions for which evaluation is higher).
>
> Nevertheless, the method as it has just been described wouldn't give good results. Indeed, since we go back to "worse" solutions, it is possible that at an iteration #j we meet a solution $\alpha_j^*$ which has already been used as $\alpha_i$ at a previous iteration #i (where $i < j$). Therefore, at iteration #(j+1), when we'll seek the "best" item of $V(\alpha_j^*)$, we'll necessarily choose the item that had been chosen at iteration #i (except if there are ex-aequos). As a result, the method would cycle and always repeat the iterations #i, #(i + 1),..., #(j - 1), #j. In practice, such a cycle would appear on very short term after having encountered the first local minimum.
>
> To avoid this cycling, the "tabu search" method lists all the items the neighbourhood of which has already been considered. So, when it has to seek $\alpha_j^*$, that is the "best" item of $V(\alpha_j)$, it won't take into consideration items which are classified as "tabu" and which could initialize cycles if they were chosen.

The "tabu search" philosophy - presented as a flow chart on next page - leads, in its very principles, to a clearly intuitive method, quite simple to implement.

In its most recent developments, the "tabu search" strategy takes into account the historical record of the search in order to adapt its process. That gives "tabu" the flexibility considered necessary by F. Glover for solving combinatorial optimization problems.

Recent implementations of "tabu search" have shown that the performance of this method can be higher than that of traditional heuristic methods.

We see on the flow chart that we have to keep a record of the best solution encountered during the search but also that we have to use a stop condition, based for example on the relative improvement of the solution during the latest iterations or on the time that has elapsed since the beginning.


Start
Choose α ∈ S
α^M = α
Stop ?
Yes
No
f* = +∞
∀ α ∈ V(α)
f(α) < f* ?
No
Yes
α tabu ?
Yes
No
α* = α
f* = f(α)
Update "tabu" list
f* < f(α^M) ?
No
Yes
α^M = α*
α = α*
Suggested sol. : α^M
Stop

## 5. Simulated annealing [5]

The "tabu search" metaheuristic, that we have just presented, was one of the improvements we could make to the "steepest descent strategy" so that the search doesn't stop at the first local minimum encountered, but continues to search a global minimum. "Simulated annealing" is another metaheuristic which aims at the same objective.

The "simulated annealing" metaheuristic results from S. Kirkpatrick, C. D. Jr. Gelatt and M. P. Vecchi's (1983) and V. Cerny's (1985) studies. They have applied the algorithm mainly to the T.S.P. Since then, the method was applied in many fields and was subject to many variants and improvements.

If "tabu search" only came into operation after the steepest descent strategy had encountered the first local minimum - before that, the simple steepest descent strategy was used - that's not the case for "simulated annealing". Indeed the "simulated annealing" metaheuristic comes into operation as soon as the search begins.

"Tabu search" can be considered as a strategy based on "willingness", "consistency" : it consists in always retaining the best solution which can be found in a neighbourhood, even if this solution is not as good as the one considered before. The "simulated annealing" metaheuristic is based on a "random" process, on "change" : in each neighbourhood considered, a solution is retained at random.

To escape from cycles, "tabu search" lists the last solution considered. "Simulated annealing" relies on its "random" approach for not considering too many times the same solutions.

After this short comparison between the main two metaheuristics, we could say "tabu search" follows a "deterministic" approach and "simulated annealing" a "probabilistic" one. This basic difference will appear very clearly if, for the same problem, we use several times the same algorithm. Indeed, although "tabu search" (in its basic variants) will always give the same results, "simulated annealing" will always retain different solutions.

The philosophy of "simulated annealing" comes directly from physics and more precisely from thermodynamics applied to metallurgy. Indeed, as the name indicates, this metaheuristic consists in simulating the annealing treatment to which metal is submitted. By means of this treatment, states which are not in thermal equilibrium - because of prior mechanical or thermal treatments - are eliminated. Annealing consists in fact in warming up the metal to a high temperature and then cooling it slowly.

Without going into too many details about the physical annealing, we could simply say that the slowliness of the cooling enables the metal to reorganize its structure during its solidification ; so, once it is solidified the metal has a new structure with minimal energy.

Then the analogy between the physical annealing process and the "simulated annealing" metaheuristic appears quite clearly.

---

5 Based on (11), (14), (20), (21), (24) and (25).

Every solution must be considered as a particular structure of metal. At each structure we can associate an energy level ; at each solution we can associate - by means of the evaluation function - a value which will represent the "energy level" of this solution. The analogy is obvious : if, in the real process, the structure tends to a minimal energy, we can hope that in the simulated process the solution will evoluate in such a way that its evaluation tends to a minimum.

When temperature is high, that is at the beginning of the process, large modifications of the structure may occur ; therefore, at the beginning of the search, we'll authorize to go easily from one solution to another, even if its energy level is higher (that is : even if its evaluation is worse). On the contrary, as temperature decreases, solidification intervenes, and structure changes not producing energy decreases are more unlikely ; as a result, the "simulated annealing" metaheuristic will gradually reduce its probability of accepting a solution with a higher "energy level".

"Simulated annealing" consists thus in choosing any solution $\alpha$ from S and in selecting at random, in a neighbourhood $V(\alpha)$ of this solution, another solution $\alpha'$. If the evaluation of that new solution is lower than that of $\alpha$, it is retained for the next iteration (and we'll consider at the next iteration the neighbourhood of $\alpha'$) ; on the contrary, if $\alpha'$ is not "better" than $\alpha$, it could be retained for the next iteration but only with a probability which decreases during the process. If $\alpha'$ is not retained for the next iteration, that iteration will consist in choosing a new solution at random in $V(\alpha)$.

As for "tabu search" we notice that we must keep a record of the best solution met during the process but also that we have to use a stop condition.

The flow chart on next page gives a more precise description of the way "simulated annealing" operates.

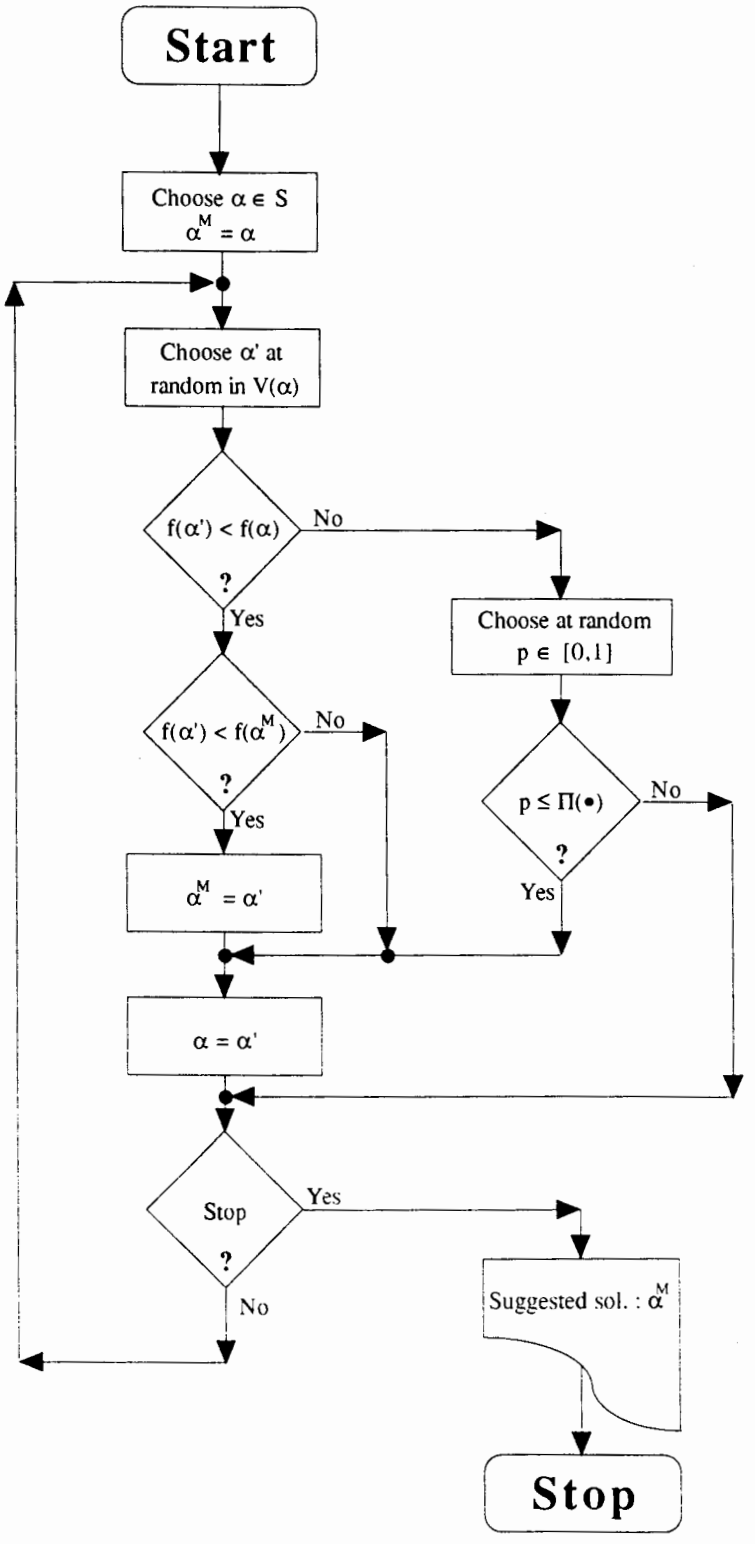
In order to make the chart clear, the probability of accepting a solution $\alpha'$ which is not better than $\alpha$ (i.e. $f(\alpha') \geq f(\alpha)$) was simply noted "$\Pi(\bullet)$". This "acceptance probability" depends on several factors ; those factors are represented by "$\bullet$". Usually they are two in number. On the one hand, the acceptance probability is an increasing function of a variable which decreases regularly during the search ; this variable is generally noted "T" by analogy with decreasing "temperature" in the metallurgic annealing process. On the other hand, the acceptance probability of $\alpha'$ is all the lower as $\alpha'$ is "bad" compared to $\alpha$ (i.e. $f(\alpha')$ is high compared to $f(\alpha)$).

Thus the acceptance probability is $\Pi(T, \Delta f)$, with :

$$\frac{\delta \Pi}{\delta T} \geq 0 \qquad \text{and} \qquad \frac{\delta \Pi}{\delta \Delta f} \leq 0$$

In accordance with thermodynamic theory, a function based on Boltzmann's distribution is usually used :

$$\Pi(T, \Delta f) = e^{-\frac{\Delta f}{T}}$$


Start
Choose α ∈ S
α^M = α
Choose α' at random in V(α)
f(α') < f(α) ?
No
Yes
Choose at random p ∈ [0,1]
f(α') < f(α^M) ?
No
Yes
p ≤ Π(•) ?
No
Yes
α^M = α'
α = α'
Stop ?
Yes
No
Suggested sol. : α^M
Stop

## 6. The T.S.P. [6]

One of the most simple dispatching problems to which we could apply "tabu search" and "simulated annealing" metaheuristics is the T.S.P. ...

Let's consider n different places 1, ..., n and let's assume a person being in another place (0) must go one time through each of the remaining n places - 1, ..., n - before coming back to his start in 0. Given the cost of the move from i directly to j (where $i, j \in \{0, ..., n\}$), the T.S.P. will consist in finding in which order the person must go through the different places to minimize the global cost of its route.

More mathematically, the terms of the T.S.P. are the following ones :

If $X = \{0, ..., n\}$ and G(X,U) is a complete and valuated graph :

$$d : U \rightarrow \mathbf{R} : (i, j) \rightarrow d(i, j) = d_{ij} \quad \text{where } i, j \in \{0, ..., n\} \ (i \neq j)$$

$$\text{with } d_{ij} \geq 0 \quad \forall\, i, j \in \{0, ..., n\} \ (i \neq j)$$

the T.S.P. consists in finding a cycle (route) $\alpha = (i_0\ i_1\ i_2\ \dots\ i_j\ \dots\ i_{n-1}\ i_n\ i_0)$ with $\{ i_j \mid j \in \{0, ..., n\} \} = \{0, ..., n\}$ such that :

$$\sum_{(i,j) \in \alpha} d_{ij} = \min_{\beta \in S} \left( \sum_{(i,j) \in \beta} d_{ij} \right)$$

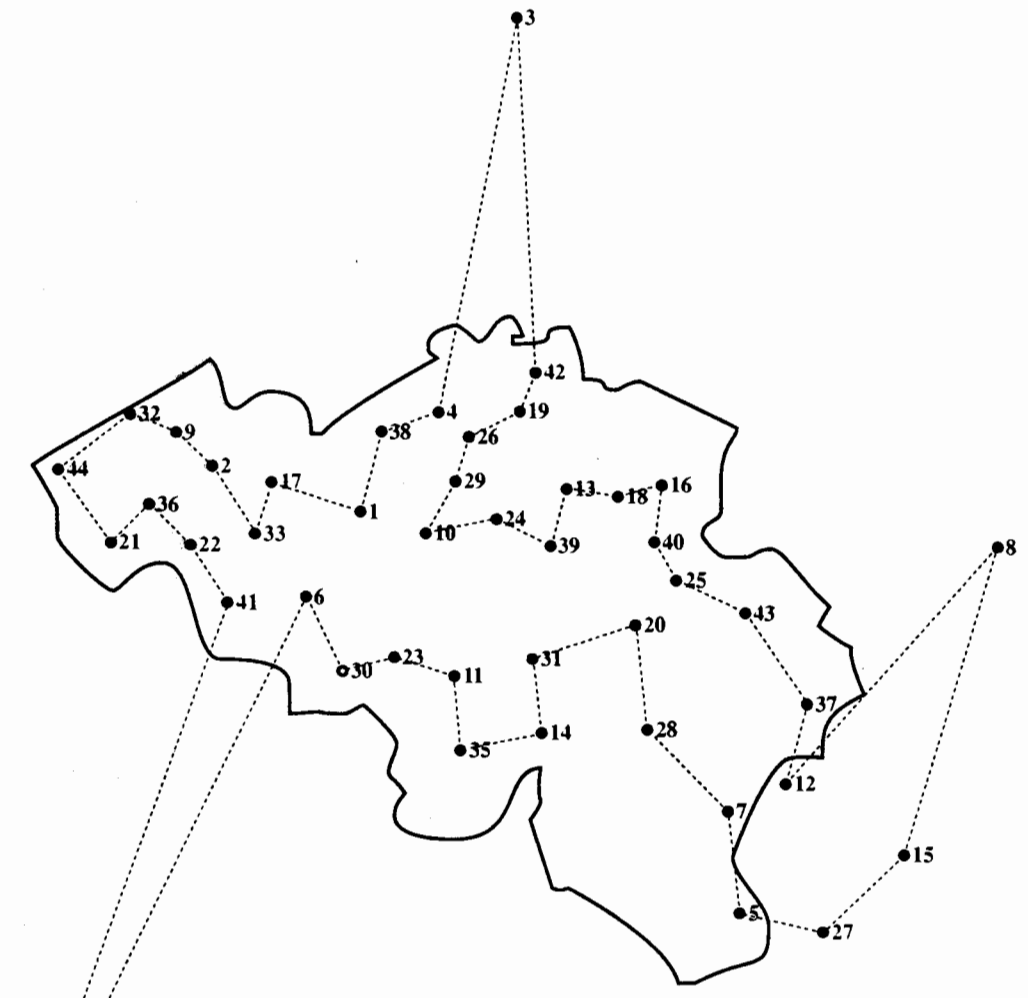
$$\text{where } S = \left\{ (i_0\, i_1\, i_2 \dots i_j \dots i_{n-1}\, i_n\, i_0) \mid \{ i_j \mid j \in \{0, ..., n\} \} = \{0, ..., n\} \right\}$$

Applying "tabu search" or "simulated annealing" in solving the T.S.P. is no big deal : we can easily build an initial solution (that's a simple permutation of vertices), consider the neighbourhood of a solution and evaluate every solution encountered. Only technical choices require a bit of "feeling" and experience. As a result, we'll only present here the best route obtained for the problem which consists in finding the shortest route going through the 44 cities presented on next page, given the distances between every two cities.

The route which is represented is the best one we obtained after several attempts in applying "tabu search" and "simulated annealing" (with a total distance of 2111 kilometers).

---

6 Based on (2), (5), (8), (12), (13) and(17).




| | | | |
|---|---|---|---|
| 1 | Aalst | 23 | La Louvière |
| 2 | Aalter | 24 | Leuven |
| 3 | Amsterdam | 25 | Liège |
| 4 | Antwerpen | 26 | Lier |
| 5 | Arlon | 27 | Luxembourg |
| 6 | Ath | 28 | Marche |
| 7 | Bastogne | 29 | Mechelen |
| 8 | Bonn | 30 | Mons |
| 9 | Brugge | 31 | Namur |
| 10 | Bruxelles | 32 | Oostende |
| 11 | Charleroi | 33 | Oudenaarde |
| 12 | Clervaux | 34 | Paris |
| 13 | Diest | 35 | Philippeville |
| 14 | Dinant | 36 | Roeselare |
| 15 | Echternach | 37 | Sankt Vith |
| 16 | Genk | 38 | Sint Niklaas |
| 17 | Gent | 39 | Tienen |
| 18 | Hasselt | 40 | Tongeren |
| 19 | Herentals | 41 | Tournai |
| 20 | Huy | 42 | Turnhout |
| 21 | Ieper | 43 | Verviers |
| 22 | Kortrijk | 44 | Veurne |

## 7. More concrete problems [7]

Although the T.S.P. is probably the most studied problem, it is quite far from daily concerns of dispatchers. Indeed, in reality, many constraints intervene and make the organization of the dispatching much more complicated : capacity constraints, schedule constraints,... Moreover, most of the time, a company has several vehicles at its disposal, and even sometimes several warehouses.

Here we see clearly how profitable metaheuristics can be. Thanks to their flexibility metaheuristics are able to solve much more concrete problems, too. Indeed, it's possible to modify the algorithm related to the T.S.P. so that specificities of a real dispatching can be taken into account. Moreover, using metaheuristics makes it possible to maintain a good level of performance provided some small adaptations of technical parameters are performed.

If we consider the T.S.P. as a reference problem - for which we have an algorithm - we can distinguish two kinds of dispatching problems :

- Problems which can be transformed into a T.S.P. and which can thus be solved by methods related to the T.S.P.

  We'll qualify these problems as being "T.S.P.-solvable".

- Problems which are different enough from the T.S.P. to make an adaptation of the algorithm inevitable.

  We'll only say about those problems that they are not "T.S.P.-solvable".

Nevertheless we must point out that "T.S.P.-solvability" is not an absolute characteristic. So, if we consider two different algorithms $A_1$ and $A_2$, built especially for solving the T.S.P., it's possible that a problem can be solved by $A_1$ but not by $A_2$. Therefore, when we speak about "T.S.P.-solvability", we should always specify which algorithm we are talking about.

As "T.S.P.-solvable" problems don't require algorithm modifications (given an algorithm for the T.S.P. of course), we won't pay more attention to them, and we'll only present the way we can adapt an initial algorithm - here we'll consider e.g. the algorithm related to the T.S.P. - for it to take extra constraints into account.

## 8. Taking extra constraints into account [8]

Let's remember what the terms of the T.S.P. are ...

---

7 Based on (4), (6), (7), (9), (19) and (26).

8 Based on (10), (13) and (21).

If $X = \{0, ..., n\}$ and $G(X,U)$ is a complete and valuated graph :

$$d : U \rightarrow \mathbf{R} : (i, j) \rightarrow d(i, j) = d_{ij} \quad \text{where } i, j \in \{0, ..., n\} \ (i \neq j)$$

$$\text{with } d_{ij} \geq 0 \quad \forall\, i, j \in \{0, ..., n\} \ (i \neq j)$$

If $S = \{ (i_0\, i_1\, i_2 \ldots i_j \ldots i_{n-1}\, i_n\, i_0) \mid \{ i_j \mid j \in \{0, ..., n\} \} = \{0, ..., n\} \}$

If f is the following function :

$$f : S \rightarrow \mathbf{R} : \beta \rightarrow f(\beta) = \sum_{(i,j) \in \beta} d_{ij}$$

Then, the T.S.P. consists in finding $\alpha \in S$ such that :

$$f(\alpha) = \min_{\beta \in S} f(\beta)$$

Let's consider now a problem derived from the T.S.P. - we'll simply call it d.T.S.P. - and obtained, starting with the T.S.P., by constraining the routes (items of S) to verify several extra constraints we'll refer to as (*).

$$\text{d.T.S.P.} = \begin{cases} \text{T.S.P.} \\ \text{Constraints (*)} \end{cases}$$

We can build S', subset of S made of items of S verifying the (*) constraints. Then, the d.T.S.P. considered consists in finding $\alpha \in S'$ such that :

$$f(\alpha) = \min_{\beta \in S'} f(\beta)$$

As the metaheuristics applied to the T.S.P. consist in finding an item of S which gives the smallest value to f, we'll "only" have here - in order to solve the d.T.S.P. - to consider items of S' instead of items of S.

In practice, the modifications will have consequences on

- The choice of an initial solution :

  The initial solution will not only have to belong to S but also to S' ; in other words, solutions which don't verify (*) must be excluded.

  Therefore we'll possibly have to build an algorithm aiming at generating items of S'. If, for the T.S.P., building an initial solution was no big deal, it's different when new constraints have been added.

When the (*) constraints only eliminate from S' few solutions of S, it is possible to choose items in S and only keep the first one which verifies (*). This approach is often much simpler but is not possible if many items of S don't belong to S'.

- The choice of a solution in the neighbourhood of another one :

  For the steepest descent strategy and for the metaheuristics we presented, we often have to consider a neighbourhood of a solution and to select a solution in this neighbourhood ; that new solution will now have to satisfy (*) . It means that we won't consider the neighbourhood V anymore but V' where $V' = V \cap S'$.

  By means of computers, if (*) only eliminates few elements of S, we may continue to scan all items of V but only select solutions satisfying the new constraints.

  With this approach we are not obliged to modify deeply the neighbourhood structure compared to what it was for the T.S.P. We only have to insert a routine checking if a specific route verifies (*) or not, which is often much easier.

  On the contrary, if (*) results in eliminating many items of S, it'll be better to examine only items which satisfy (*), so that the computation time doesn't increase too much. In such a case, we'll have to build a new neighbourhood structure which is such that :

  $$V' : \alpha \rightarrow V'(\alpha) \qquad \text{where } \alpha \in S' \text{ et } V'(\alpha) \subset S'$$

  Mathematically considered, this can prove to be much more complicated than checking if a specific solution satisfies the (*) constraints or not.

As we can see, if the (*) constraints are strong - we mean that they exclude from S' many items of S -, we have to modify quite considerably the way the metaheuristics operate ... or computation time will increase drastically ; those modifications require a good mathematical background about differences generated by the new constraints. Moreover, technical parameters will have to be more adapted than for slight modifications of the algorithm.

Therefore, solving the d.T.S.P. with strong constraints instead of the T.S.P. requires a much more important investment : it is thus in opposition with the philosophy of the metaheuristics.

As a result, we'd like to focus on a possibility to bypass the problem, which can be very useful in some respects ; it consists in modifying the evaluation function instead of the space of admissible solutions.

This alternative consists in solving the problem "on" S (instead of "on" S') and in penalizing the value given by the evaluation function to items of S which don't satisfy the (*) constraints. So, for our minimization problem, we won't evaluate solutions of S by function f but well by function f ' that we'll define as follows :

$$f' : S \rightarrow \mathbf{R} : \beta \rightarrow f'(\beta) = f(\beta) + w(\beta)$$

where w is the penalty function and is such that :

$$w : S \rightarrow \mathbf{R} : \beta \rightarrow w(\beta) \begin{cases} > 0 & \text{if } \beta \notin S' \\ = 0 & \text{if } \beta \in S' \end{cases}$$

This way of adapting the algorithm is much more intuitive - and therefore more understandable - than the previous one ; that's a first advantage of it.

This method also presents other advantages :

- It makes it possible to adapt only slightly the metaheuristics, compared to what they were when trying to solve the T.S.P.

  Moreover, modifications will rarely imply more than simply adding some little modules to the initial algorithm ; each of these modules consisting in computing the penalty (or part of the penalty) to be applied to f.

- It does not force us to build solutions of S' really, as it works with all solutions of S. This advantage is very important when building a solution of S' implies the use of a specific algorithm, which can possibly be **NP**-hard. In some extreme situations, building an item of S' can even turn out to be impossible.

- It makes it possible for the algorithm to transit through not admissible areas of the space of solutions to reach admissible areas, which can possibly include better solutions.With the previous way of adapting the algorithm - way based on the space of solutions - it was not possible. Therefore, the algorithm could only go from one admissible solution to another by transiting through admissible solutions. This could slow down the algorithm in its search for better solutions.

Nevertheless some difficulties appear when we consider penalizing the function instead of reducing the space of admissible solutions :

- As #S > #S', working "on" S instead of "on" S' can considerably increase the computation time.

- We must properly choose w because too small a penalty lets the algorithm select not admissible solutions and too high a penalty cancels the last-mentioned advantage.

  In a study made in 1991, M. Gendreau, A. Hertz and G. Laporte suggested to adapt the penalty during the solving process, so that the algorithm doesn't consider too high a proportion of admissible solutions - we should *ceteris paribus* reduce the penalty - or, on the contrary, too low a proportion of admissible solutions - we should then *ceteris paribus* increase the penalty.[9]

  Moreover, w must be such that the search is guided efficiently to admissible solutions ; with this aim in view, we'll prefer variable penalties to constant penalties.

---

9 (13), 6-8.

Finally we'd like to mention that this way of adapting the algorithm enables us, by choosing proper penalties, to authorize some constraints violations if it's worthwhile ; when the function is penalized in this way, we talk about "soft constraints". Let's take the example of time windows constraints : in this case, it'll be possible to use a penalty which increases according to the advance or the delay compared to the "opening hours" (that's "time windows boundaries"). This will enable us to avoid going a long way round for only two minutes delay ; that's very important as it's not even possible in practice to foresee the time which is exactly required for a route.

After having shown how it was possible to take new constraints into consideration in the algorithms, we'll focus on the modifications needed for solving the m.T.S.P.T.W. This not "T.S.P.-solvable" variant of the T.S.P. is only an example among many other possibilities, but in practice it's probably one of the most usual.

## 9. The m.T.S.P.T.W. [10]

The m.T.S.P.T.W. is a problem - derived from the T.S.P. - which considers more than one vehicle and takes "time windows" into consideration. By "time windows" we mean periods of time during which it's possible to supply the delivery points (shops, clients,...) ; out of the "time windows" assigned to it, a delivery point is considered as not accessible.

There are many possible configurations for "time windows" and the choice will essentially depend on the practical situation the algorithm has to "solve". We'll choose here a variant of the configuration presented by M. Desrochers in (6)[11], but the way we'll introduce modifications to be applied can easily be transposed to other kinds of "time windows" constraints.

We'll consider here that a "time window" $[o_j', f_j]$ $(o_j' < f_j)$ - where $o_j'$ could be the opening hour and $f_j$ the closing hour ($j \in \{1, ..., n\}$) - is associated to each delivery point (shop, client,...).

To each arc (i, j) (where $i, j \in \{0, ..., n\}$ $(i \neq j)$) we'll associate the time $t'_{i\,j}$ which is necessary for going from i to j ...

$$t' : U \rightarrow \mathbb{R} : (i, j) \rightarrow t'(i, j) = t'_{i\,j}$$

$$\text{where } i, j \in \{0, ..., n\}\ (i \neq j)$$

We'll note $s_j$ ($j \in \{1, ..., n\}$) the time which is necessary for the "service" at delivery point j ; this "service" is the unloading of goods in a very large sense. $s_j$ corresponds thus to the time that every vehicle must spend in j. We'll agree that $s_0 = 0$.

Let's assume that "time windows" and "service" time spans are compatible, that is :

$$f_j - o_j' \geq s_j \qquad \forall\, j \in \{1, ..., n\}$$

---

10 Based on (6).
11 (6), 66-67.

We'll include here the "service" time at stop j ($j \in \{0, ..., n\}$) in the time needed for going from i to j ($i \in \{0, ..., n\}\backslash\{j\}$) ; therefore we'll evaluate the time needed for going from i to j by $t_{ij}$ instead of $t'_{ij}$ :

$$t : U \rightarrow \mathbf{R} : (i, j) \rightarrow t(i, j) = t_{ij} = t'_{ij} + s_j$$

$$\text{where } i, j \in \{0, ..., n\} \ (i \neq j)$$

If $A_j$ is the arrival time in j and $D_j$ is the time the vehicle leaves j, we'll have to check the following constraints, which we'll refer to as $\mathcal{C}$ :

- We'll consider that there can't be any delay after the unloading and that the vehicle departure must occur, at last, when the delivery point closes its doors. This is not restrictive as we'll authorize a delay before the unloading.

  $$D_j \leq f_j \qquad (j \in \{1, ..., n\})$$

  The unloading may only begin after the opening of the delivery point. As the unloading in j requires a time $s_j$, we must satisfy the following constraint :

  $$D_j - s_j \geq o_j' \quad (j \in \{1, ..., n\})$$

  or $\quad D_j \geq o_j \qquad (j \in \{1, ..., n\})$

  where $o_j$ (with $o_j = o_j' + s_j$) is the moment before which the departure can't take place, given the time $s_j$ which is necessary for the delivery.

  Finally :

  $$o_j \leq D_j \leq f_j \quad (j \in \{1, ..., n\})$$

  We'll notice that there are no "time windows" constraints for the warehouse as it is assumed to be opened when it's necessary.

- The time between the moment a vehicle leaves a point i ($i \in \{0, ..., n\}$) and the moment the same vehicle leaves the next point in its route ($\beta$) - let's call it j ($j \in \{0, ..., n\}\backslash\{i\}$) - can't be less than the sum of the time needed for the drive ($t'_{ij}$) and of the time needed for the unloading in j ($s_j$). As a result, we must have :

  $$D_j - D_i \geq t'_{ij} + s_j \qquad (i, j \in \{0, ..., n\} \mid (i, j) \in \beta)$$

  or $\quad D_j - D_i \geq t_{ij} \qquad (i, j \in \{0, ..., n\} \mid (i, j) \in \beta)$

  or $\quad D_i + t_{ij} \leq D_j \qquad (i, j \in \{0, ..., n\} \mid (i, j) \in \beta)$

As we exclude any "time windows" constraints for the warehouse and the vehicles, $D_0$ may be fixed unrestrainedly. Thus, the constraints may be expressed as :

$$D_i + t_{ij} \leq D_j \qquad (i, j \in \{1, ..., n\} \mid (i, j) \in \beta)$$

Given these constraints, we notice that :

$$A_j = D_i + t'_{ij} = D_i + t_{ij} - s_j \leq D_j - s_j$$

$$(i, j \in \{1, ..., n\} \mid (i, j) \in \beta)$$

Thus : $A_j + s_j \leq D_j \qquad (i, j \in \{1, ..., n\} \mid (i, j) \in \beta)$

As a result, we note that the arrival time in j ($A_j$) is not necessarily equal to ($D_j$ - $s_j$) ; so, we authorize the vehicle to stay in j longer than it would be strictly necessary for the unloading of goods. Owing to that, we can consider the case where a vehicle comes in j before $o_j'$, i.e. before the opening time of the delivery point ; in such a case, a delay ($o_j'$ - $A_j$) is imposed to the vehicle.

These terms which authorize a delay are of great interest in practice as they avoid some absurdities in the results obtained. Indeed, if no delay is authorized, the algorithm is sometimes bound to have the vehicle go a long way round for the only reason it would otherwise arrive at the delivery point two minutes before its opening.

After this small "theoretical" presentation of the terms chosen here for the m.T.S.P.T.W., we'll explain the way the metaheuristic can be modified to take the new constraints into account.

We must first notice that introducing "time windows" limits the number of delivery points which can be visited during a single route. As we don't consider here the case of a same vehicle making several routes - but rather the case of every vehicle making one and only one route - we'll of course have to consider the existence of several vehicles in 0.

One of the problems which will appear consists in determining how many vehicles we must foresee. If too many vehicles are foreseen the algorithm won't assign them to a route and, if no vehicles enough are foreseen, the algorithm won't be able to find an admissible solution. As we can see, the best would be to choose a number of vehicles which is sufficient with a 100 % probability. Nevertheless we must take into consideration the fact that introducing more vehicles involves increasing the time needed for solving the problem. Therefore we advise to foresee a sufficient number of vehicles but without exaggeration.

When, in a real situation, the company owns a certain number of vehicles we can assume that this fleet is reasonable. Therefore, in an operational or tactical framework - in opposition to a strategic one - we'll choose the number of vehicles we really have at our disposal.

In order to take into account the existence of k vehicles ($k > 1$) at the warehouse, we'll use a stratagem which consists in considering that each of the k vehicles is in a different warehouse ...

Let's nickname 0.1 the vertex corresponding to the warehouse and let's name 0.i ($i = 2, ..., k$) the ($k - 1$) vertices related to the fictitious warehouses. Note that 0.1 is only a nickname and

that we may refer to the same vertex with 0, when we consider all the real vertices (warehouse or delivery points), or with 0.1, when we consider all the warehouses (fictitious or real).

Nevertheless, as all vehicles are located in the same place, we'll consider that the fictitious warehouses have the same "relations" as the real warehouse ; we mean :

$$d_{0.i\,j} = d_{0.1\,j} \qquad \forall\, i \in \{2, ..., k\}\,; \forall\, j \in \{1, ..., n\}$$
$$d_{j\,0.i} = d_{j\,0.1} \qquad \forall\, i \in \{2, ..., k\}\,; \forall\, j \in \{1, ..., n\}$$
$$d_{0.i\,0.j} = 0 \qquad \forall\, i \in \{1, ..., k\}\,; \forall\, j \in \{1, ..., k\} \setminus \{i\}$$

The stratagem consisting in creating fictitious warehouses will then lead us to add (k - 1) rows and (k - 1) columns to the matrix relating to the problem. Since the fictitious warehouses have the same "relations" as the real warehouse, these new rows and columns will of course be identical respectively to the row and the column related to stop 0.

In the solution obtained, each time the route will meet a (real or fictitious) warehouse, it will have to be considered as the end of the present route (related to one of the vehicles) and (possibly) the start of the next vehicle, for another route.

To be closer to reality, it's possible to integrate a fixed cost for the use of every extra vehicle. If this cost is called c, it'll be taken into account by adding it to each $d_{1.p\,j}$ (where $p \in \{1, ..., k\}$, $j \in \{1, ..., n\}$). In other words, we'll add c to each of the last n items of the first k rows of the value matrix.

To incorporate the new constraints we've just presented - concerning the m.T.S.P.T.W. - into the algorithm related to T.S.P., we'll use the function-based approach. So, we'll avoid having to build an initial solution satisfying the $\mathbf{C}$ constraints ; we'll be able to begin with a solution which violates one or more new constraints.

The objective function used is the following one :

$$f' : S \to \mathbb{R} : \beta \to f'(\beta) = f(\beta) + w(\beta)$$

where w is the penalty function and is such that :

$$w : S \to \mathbb{R} : \beta \to w(\beta) \begin{cases} > 0 & \text{if } \beta \text{ doesn't verify } \mathbf{C} \\ = 0 & \text{if } \beta \text{ verifies } \mathbf{C} \end{cases}$$

Although the matrix to which the algorithm will be applied can be symmetric, it's important to notice that the problem is not symmetric. Indeed, if $\beta$ is a route and $\beta'$ is its symmetric, we may have $w(\beta) \neq w(\beta')$ and thus $f'(\beta) \neq f'(\beta')$. The algorithm can't neglect then to consider both $\beta$ and $\beta'$.

To check if a solution $\beta = (i_0\ i_1\ i_2 \ldots i_j \ldots i_{n-1}\ i_n\ i_0)$ of S ( where $\{\ i_j \mid j \in \{0,\ldots, n\}\ \} = \{0, \ldots, n\}$ ) verifies **C** or not, we'll need :

- $o_i$ and $f_i$ ( $i \in \{1, \ldots, n\}$ )
- $t_{i\,j}$ ( $i,j \in \{1, \ldots, n\}$ $(i \neq j)$ )

For convenience sake, the $t_{i\,j}$ will be presented as the $d_{i\,j}$ in the form of a $(n + 1) \times (n + 1)$ matrix called "time matrix". The transformation applied to the "distance matrix", in order to take account of the fact that there are k vehicles in 0, will also have to be applied to the "time matrix". Therefore, we'll finally have two $(n + k) \times (n + k)$ matrices.

Let's repeat that the **C** constraints are the following ones :

$$o_j \leq D_j \leq f_j \qquad (j \in \{1, \ldots, n\}) \qquad (\mathbf{C}\ 1)$$

$$D_i + t_{i\,j} \leq D_j \qquad (i, j \in \{1, \ldots, n\} \mid (i, j) \in \beta) \qquad (\mathbf{C}\ 2)$$

As a result, for the route $\beta \in S$, we'll have to compute the $D_i$ ( $i \in \{1, \ldots, n\}$ ) related to this route ; this computation will be based on **C** 2 but we'll try to satisfy **C** 1 as good as possible. Then we'll compute $w(\beta)$ on the basis of the violations of **C** 1.

Let's consider $p \in \{1, \ldots, k\}$ and $j \in \{1, \ldots, n\}$ such that $(0.p, j) \in \beta$.

**C** 2 doesn't impose any lower limit to $D_j$. Thus, we should fix $D_j = o_j$.

Let's consider $p \in \{1, \ldots, k\}$ and $j \in \{1, \ldots, n\}$ such that $(j, 0.p) \in \beta$.

**C** 2 doesn't impose any upper limit to $D_j$. Thus, we'll be able to "push" $D_j$ until $D_j = f_j$.

Let's consider $p \in \{1, \ldots, k\}$ and $j \in \{1, \ldots, n\}$ such that $(0.p, j) \in \beta$ and $(j, 0.p) \in \beta$.

**C** 2 doesn't impose any (lower or upper) limit to $D_j$. Thus, we'll fix $D_j$, at our own convenience, in $[o_j, f_j]$.

One of the ways we can check if constraints are satisfied is the following one :

- For every $j \in \{1, \ldots, n\}$ such that

  $\exists\ p \in \{1, \ldots, k\}$ such that $(0.p, j) \in \beta$

  we fix $D_j = o_j$.

- For every $j \in \{1, \ldots, n\}$ such that

  $\exists\ i \in \{1, \ldots, n\}$ such that $\begin{cases} (i, j) \in \beta \\ D_i \text{ has already been computed} \end{cases}$

we fix $D_j = \max ( D_i + t_{ij} ; o_j )$

We repeat this step until $D_j$ has been computed for all $j \in \{1, ..., n\}$.

Practically, this second stage can be linked to the first one : from a vertex which represents a (real or fictitious) warehouse we examine all vertices in the order in which they appear in $\beta$.

- We compute $w(\beta)$ by checking if $D_i \in [o_i, f_i], \forall i \in \{1, ..., n\}$.

If we don't accept any violation of "opening hours", we'll use for example :

$w(\beta) = 0$ if $D_i \leq f_i, \forall i \in \{1, ..., n\}$

$w(\beta) = M$ if $\exists i \in \{1, ..., n\} \mid D_i > f_i$

(where M is such that $M > r \quad \forall r \in \mathbf{R}$ to which M is compared)

If we judge that one unit of time delay costs the same amount to the company as an increase by A of the sum of values of the routes, we'll use for example :

$$w(\beta) = A \cdot \sum_{i=1}^{n} \max ( D_i - f_i : 0 )$$

We may of course imagine many other possibilities for the penalty : penalty with exponential increase,...

Nevertheless it's important to choose a kind of penalty which is compatible with the logic of the algorithm. For an enumeration algorithm an "all or nothing" penalty can perfectly be valid ; for metaheuristics - which don't try all solutions - it's better to use a variable penalty, so that the search is guided to solutions which satisfy "time windows" constraints.

Finally, when a solution has been obtained, we can set up the planning of departures and arrivals as follows :

Let's consider $j \in \{1, ..., n\}$ :

$D_j$ is computed as mentioned above.

If $\exists i \in \{1, ..., n\}$ such that $(i, j) \in \beta$ :

$A_j = D_i + t'_{ij}$

Otherwise :

$$A_j = D_j - s_j$$

Let's consider $p \in \{1, ..., k\}$ :

$$D_{0.p} = A_j - t'_{0.p\,j} = o_j - s_j - t'_{0.p\,j} = o_j - t_{0.p\,j}$$

where $j \in \{1, ..., n\}$ is such that $(0.p, j) \in \beta$

If we can't find such a j, $D_{0.p}$ can't be computed.

$$A_{0.p} = D_j + t'_{j\,0.p} = D_j + t_{j\,0.p}$$

where $j \in \{1, ..., n\}$ is such that $(j, 0.p) \in \beta$

If we can't find such a j, $A_{0.p}$ can't be computed.

We repeat the same process until the entire planning has been set up. As we already mentioned it, we may set up the planning by examining only once the vertices of $\beta$, provided that these vertices are examined in sequence order, starting at a vertex representing a warehouse.

As an example, let's consider the continuation of the problem presented for the T.S.P.

As far as "time matrix" is concerned, we'll assume that every vehicle has an average speed of 60 km/h, that is 1 min/km. As far as "time windows" are concerned, the next page lists the opening and closing time considered for each delivery point.

We used the following function for the penalty related to "time windows" violations :

$$w : S \rightarrow \mathbb{R} : \beta \rightarrow w(\beta) = 4 . \sum_{i=1}^{n} \max ( D_i - f_i ; 0 )$$

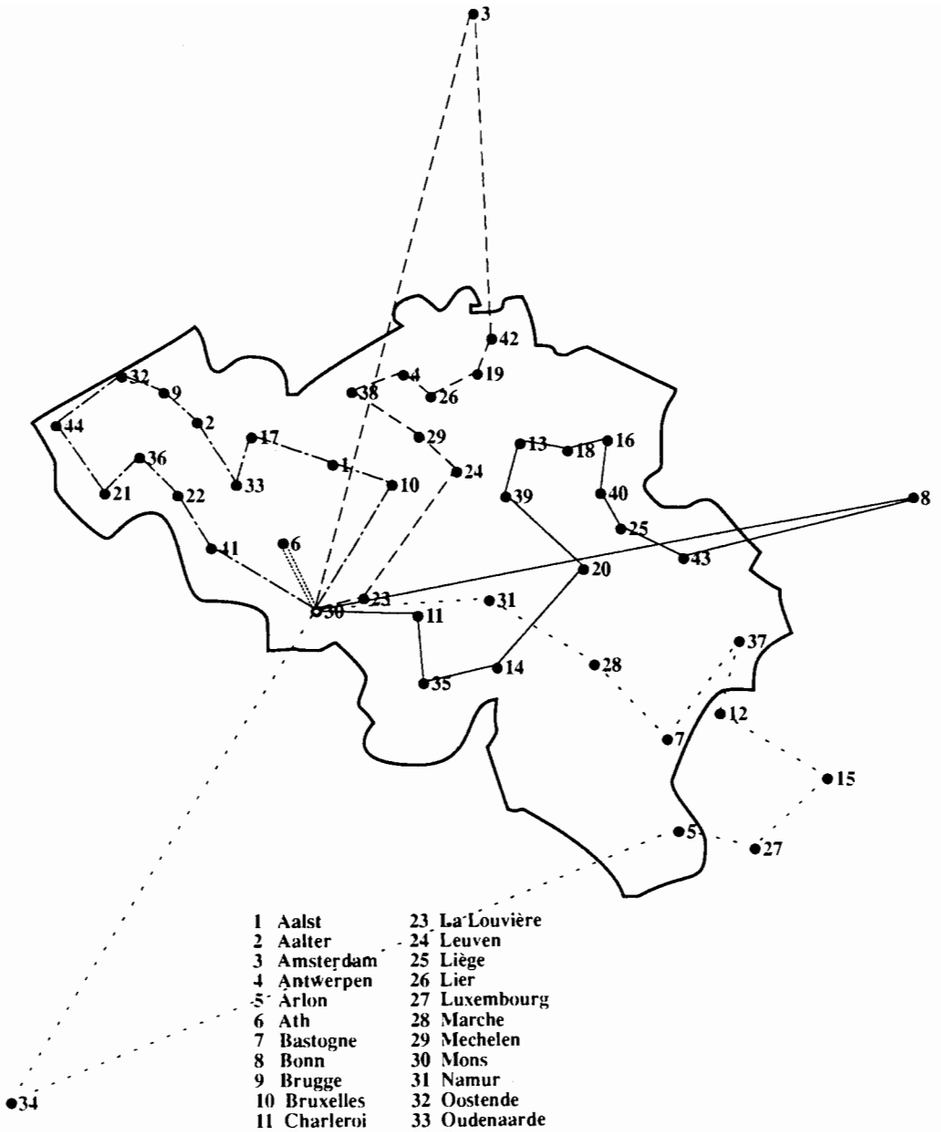
We'll assume that the index i = 0 refers to the warehouse, for every vehicle considered.

By using such a function, we consider that the cost for the company is the same for one minute's delay as for 4 extra kilometers. So, we did not definitely forbid violations of time constraints.

Let's finally mention that we used the minute as time unit, that the "service" time has been fixed at 15 minutes, for all delivery points, and that we have considered that there were 10 vehicles available in the warehouse of Mons.

The best result obtained from the 10 tests performed is represented on the map : this solution is 2704 kilometers long and there are no violations of "time windows" constraints. Only 5 out of the 10 vehicles were used.

| Delivery point | o' (hours) | o (hours) | f (hours) | o (minutes) | f (minutes) |
|---|---|---|---|---|---|
| *Aalst* | 8.00 | 8.15 | 16.00 | 495 | 960 |
| *Aalter* | 9.00 | 9.15 | 14.00 | 555 | 840 |
| *Amsterdam* | 7.00 | 7.15 | 20.00 | 435 | 1200 |
| *Antwerpen* | 7.00 | 7.15 | 18.00 | 435 | 1080 |
| *Arlon* | 8.00 | 8.15 | 16.00 | 495 | 960 |
| *Ath* | 9.00 | 9.15 | 14.00 | 555 | 840 |
| *Bastogne* | 8.00 | 8.15 | 16.00 | 495 | 960 |
| *Bonn* | 7.00 | 7.15 | 20.00 | 435 | 1200 |
| *Brugge* | 8.00 | 8.15 | 16.00 | 495 | 960 |
| *Bruxelles* | 7.00 | 7.15 | 20.00 | 435 | 1200 |
| *Charleroi* | 8.00 | 8.15 | 16.00 | 495 | 960 |
| *Clervaux* | 9.00 | 9.15 | 14.00 | 555 | 840 |
| *Diest* | 9.00 | 9.15 | 14.00 | 555 | 840 |
| *Dinant* | 8.00 | 8.15 | 16.00 | 495 | 960 |
| *Echternach* | 9.00 | 9.15 | 14.00 | 555 | 840 |
| *Genk* | 9.00 | 9.15 | 14.00 | 555 | 840 |
| *Gent* | 7.00 | 7.15 | 18.00 | 435 | 1080 |
| *Hasselt* | 8.00 | 8.15 | 16.00 | 495 | 960 |
| *Herentals* | 9.00 | 9.15 | 14.00 | 555 | 840 |
| *Huy* | 9.00 | 9.15 | 14.00 | 555 | 840 |
| *Ieper* | 9.00 | 9.15 | 14.00 | 555 | 840 |
| *Kortrijk* | 8.00 | 8.15 | 16.00 | 495 | 960 |
| *La Louvière* | 9.00 | 9.15 | 14.00 | 555 | 840 |
| *Leuven* | 9.00 | 9.15 | 14.00 | 555 | 840 |
| *Liège* | 7.00 | 7.15 | 18.00 | 435 | 1080 |
| *Lier* | 9.00 | 9.15 | 14.00 | 555 | 840 |
| *Luxembourg* | 7.00 | 7.15 | 20.00 | 435 | 1200 |
| *Marche* | 8.00 | 8.15 | 16.00 | 495 | 960 |
| *Mechelen* | 9.00 | 9.15 | 14.00 | 555 | 840 |
| *Mons (warehouse)* | - | - | - | - | - |
| *Namur* | 7.00 | 7.15 | 18.00 | 435 | 1080 |
| *Oostende* | 8.00 | 8.15 | 16.00 | 495 | 960 |
| *Oudenaarde* | 9.00 | 9.15 | 14.00 | 555 | 840 |
| *Paris* | 7.00 | 7.15 | 20.00 | 435 | 1200 |
| *Philippeville* | 9.00 | 9.15 | 14.00 | 555 | 840 |
| *Roeselaere* | 8.00 | 8.15 | 16.00 | 495 | 960 |
| *Sankt Vith* | 9.00 | 9.15 | 14.00 | 555 | 840 |
| *Sint Niklaas* | 9.00 | 9.15 | 14.00 | 555 | 840 |
| *Tienen* | 9.00 | 9.15 | 14.00 | 555 | 840 |
| *Tongeren* | 8.00 | 8.15 | 16.00 | 495 | 960 |
| *Tournai* | 8.00 | 8.15 | 16.00 | 495 | 960 |
| *Turnhout* | 8.00 | 8.15 | 16.00 | 495 | 960 |
| *Verviers* | 7.00 | 7.15 | 18.00 | 435 | 1080 |
| *Veurne* | 9.00 | 9.15 | 14.00 | 555 | 840 |




| | | | |
|---|---|---|---|
| 1 | Aalst | 23 | La Louvière |
| 2 | Aalter | 24 | Leuven |
| 3 | Amsterdam | 25 | Liège |
| 4 | Antwerpen | 26 | Lier |
| 5 | Arlon | 27 | Luxembourg |
| 6 | Ath | 28 | Marche |
| 7 | Bastogne | 29 | Mechelen |
| 8 | Bonn | 30 | Mons |
| 9 | Brugge | 31 | Namur |
| 10 | Bruxelles | 32 | Oostende |
| 11 | Charleroi | 33 | Oudenaarde |
| 12 | Clervaux | 34 | Paris |
| 13 | Diest | 35 | Philippeville |
| 14 | Dinant | 36 | Roeselare |
| 15 | Echternach | 37 | Sankt Vith |
| 16 | Genk | 38 | Sint Niklaas |
| 17 | Gent | 39 | Tienen |
| 18 | Hasselt | 40 | Tongeren |
| 19 | Herentals | 41 | Tournai |
| 20 | Huy | 42 | Turnhout |
| 21 | Ieper | 43 | Verviers |
| 22 | Kortrijk | 44 | Veurne |

## 10. Core-satellites

As we have just seen it, the transformation of the algorithm aiming at solving the T.S.P. to an algorithm aiming at solving the m.T.S.P.T.W. finally amounts to adding two modules :

- one which modifies the initial matrix to create "fictitious warehouses" ;
- another one which, for every solution considered, computes a penalty based on extra data available. Thereafter, we only have to introduce the penalty in each evaluation.

It is easy to understand that incorporating other specificities than "multiple vehicles" (m.) or "time windows" (T.W.) will also amount to adding one or more specific modules to the initial algorithm. This way, we'll be able to build complex algorithms, by adding to a basic algorithm - called the "core" - some modules - called "satellites" - by means of which extra specificities of the problem may be taken into account.

Therefore, this approach - that we could call "core-satellites" approach - combined with metaheuristics - that can be adapted to solve a large range of problems - enables us to build algorithms which can be used as efficient tools for the most complex dispatching problems that can occur in reality.

## 11. Data-processing [12]

In organizing distribution routes, theoretical and practical aspects are hardly compatible :

- practical situations require taking many objectives (minimum of cost, best distribution of tasks,...) and constraints (capacity, schedules,...) into consideration ;

- from the theoretical point of view, problems considered are much too standardized and simplified. Moreover, computation time rapidly becomes excessively long even when using heuristic methods and computers.

Using metaheuristics and the "core-satellites" approach already remedies a little this relative incompatibility between theory and practice.

Yet, only data-processing can really make theory and practice closer to each other. Indeed, using computers enables us to take more constraints and objectives into account in a reasonable time. Considering existing algorithms makes it clear that only computers enable us to use them.

Nevertheless, data-processing can't replace human beings in organizing delivery routes. Indeed, even if using metaheuristics and what we called the "core-satellites" approach enables us to model the algorithm on the real situation, it is clear that in reality many reasons make it difficult to "systematize" dispatching (by means of computers) :

---

12 Based on (1) and (18).

- In practice, many human, social,... aspects - which can't possibly be mathematized - can reduce the space of possible solutions.

  For example, there are often character "incompatibilities" between lorry drivers and shop managers. Even if it's not really necessary to take this kind of aspects into account, it's better to do it, to improve relationships, motivation and thus profitability in the company.

  Other examples are drivers' preferences, respect of a fair distribution of tasks,...

- The "mathematical" respect of some constraints can be injudicious in practice ; especially when dispatchers may take liberties in organizing the routes.

  For example, for "time windows" constraints, the computer will try to satisfy exactly the constraints imposed by the user. Yet, the solution given by the computer can be not as good as the one which would have been found by a "human" dispatcher, who can give a call to the staff of a shop to ask them to be there 10 minutes earlier or later.

  For capacity constraints, a "human" dispatcher can also get a better solution by deciding that some goods, less urgent than the other ones, will be delivered the day after.

- Some specific unforeseen constraints can appear that the algorithm can't handle. The company can for example get a peculiar order for goods which require some special conditions for their transportation.

- Solutions given by mathematical algorithms can fail to take into account some "irrational" standards that human dispatchers could have. In such a situation, even if the solution obtained is good, it can be refused ; although the algorithm is efficient, it will thus appear as being inadequate.

  The "opposition to change" of the different parties concerned by the dispatching can also play a part in the failure of the computerization.

- And so on.

## 12. Conclusion

As we can see, it would be dangerous to believe that we only have to choose a mathematical algorithm to solve all dispatching problems. Nevertheless we shouldn't either conclude that the computerization of company dispatching is always a hazardous task which can only give bad results. The choice between a manual dispatching and a computerized dispatching must be the result of a global and quantified thought which takes account of the different aspects of the company organization.

If we decide to use data-processing, the combination of metaheuristics and the "core-satellites" approach can be very helpful. Indeed, through their high flexibility, these two approaches put together make it possible to solve specific problems which are far from "theoretical" standardized problems like T.S.P., V.R.P.....

## References


*(1)* Bodin L., Golden B., "Vehicle Routing Software for Microcomputers : a Survey", *in* Gass S.I., Greenberg H.J., Hoffman K.L., Langley R.W., "Impacts of Microcomputers on Operations Research", North-Holland (1985).

*(2)* Boël O., "Optimisation des Tournées de Distribution", mémoire d'étudiant, Faculté polytechnique de Mons (1984), 1-6.

*(3)* Cao B., Uebe G., "Solving transportation problem with nonlinear side constraints with tabu search", Discussions Papers in Statistics and Quantitative Economics, #46, Universität der Bundeswehr Hamburg (May 1991), 1-4.

*(4)* Christofides N., "Vehicle Routing", *in* Lawler E.L., Lenstra J.K., Rinnooy Kan A.H.G., Shmoys D.B., "The Traveling Salesman Problem", John Wiley & Sons (1985), 431-435.

*(5)* Dehem V., Du Bus B., "Problème du Voyageur de Commerce : Application à 67 Villes", mémoire d'étudiant, Université Catholique de Louvain (August 1991), 1-10.

*(6)* Desrochers M., Lenstra J.K., Savelsbergh M.W.P., Soumis F., "Vehicle Routing with Time Windows : Optimization and Approximation", *in* Golden B.L., Assad A.A., "Vehicle Routing : Methods and Studies", North-Holland (1988), 65-67.

*(7)* Desrochers M., Lenstra J.K., Savelsbergh M.W.P., "A classification scheme for vehicle routing and scheduling problems", *in* European Journal of Operational Research, volume 46 (1990), 322-324.

*(8)* Desrochers M., Laporte G., "Improvements and extensions to the Miller-Tucker-Zemlin subtour elimination constraints", *in* Operations Research Letters, volume 10 (1991), 27-28.

*(9)* Desrochers M., Verhoog T.W., "A New Heuristic for the Fleet Size and Mix Vehicle Routing Problem", *in* Computers and Operations Research, volume 18 (1991), 263-264.

*(10)* Desrochers M., Desrosiers J., Solomon M., "A New Optimization Algorithm for the Vehicle Routing Problem with Time Windows", *in* Operations Research, volume 40 (March-April 1992), 342-343.

*(11)* Eglese R.W., "Simulated Annealing : A tool for Operational Research", *in* European Journal of Operational Research, volume 46 (1990), 271-281.

*(12)* Gendreau M., Hertz A., Laporte G., "New Insertion and Post-Optimization Procedures for the Traveling Salesman Problem", publication #708, Centre de recherche sur les transports, Université de Montréal (February 1991), 1-4.

*(13)* Gendreau M., Hertz A., Laporte G., "A Tabu Search Heuristic for the Vehicle Routing Problem", publication #777, Centre de recherche sur les transports, Université de Montréal (June 1991).

*(14)* Golden B.L., Skiscim C.C., "Using Simulated Annealing to Solve Routing and Location Problems", Working Paper Series MS/S 85-010, College of Business and Management, University of Maryland (April 1985), 1-8, 28-31.

*(15)* Haouari M., Dejax P., Desrochers M., "Les Problèmes de Tournées avec Contraintes de Fenêtres de Temps, l'Etat de l'Art", *in* Recherche opérationnelle/Operations Research, volume 24 (1990), 217-219.

*(16)* Hillier F.S., Lieberman G.J., "Introduction to Operations Research", McGraw-Hill (1990), 209-212, 336-339.

*(17)* Hoffman A.J., Wolfe P., "History", *in* Lawler E.L., Lenstra J.K., Rinnooy Kan A.H.G., Shmoys D.B., "The Traveling Salesman Problem", John Wiley & Sons (1985), 9-15.

*(18)* Hooban J.M., "Marketing a Vehicle Routing Package", *in* Golden B.L., Assad A.A., "Vehicle Routing : Methods and Studies", North-Holland (1988), 447-468.

*(19)* Lenstra J.K., Rinnooy Kan A.H.G., "Complexity of Vehicle Routing and Scheduling Problems", Report 7919/0, Econometric Institute, Erasmus University of Rotterdam (1979).

*(20)* Pirlot M., "General heuristics in Combinatorial Optimization", Seminar Course, Politecnico di Torino (October-November 1991), I.1-I.28, II.10-II.13, IV.1-IV.5, V.2-V.4.

*(21)* Pirlot M., "General local search heuristics in Combinatorial Optimization : a tutorial", *in* Belgian Journal of Operations Research, Statistics and Computer Science, volume 32 (1st semester 1992), 9-47, 60-61.

*(22)* Pureza V.M., França P.M., "Vehicle Routing Problems Via Tabu Search Metaheuristic", publication #747, Centre de recherche sur les transports, Université de Montréal (January 1991).

*(23)* Rousseau J.M., "Customization Versus a General Purpose Code for Routing and Scheduling Problems : a Point of View", *in* Golden B.L., Assad A.A., "Vehicle Routing : Methods and Studies", North Holland (1988), 469-475.

*(24)* Séguin R., "Les Algorithmes de Recuit simulé en Optimisation combinatoire", mémoire d'étudiant, publication #595, Centre de recherche sur les transports, Université de Montréal (May 1988).

*(25)* Skiscim C.C., Golden B.L., "Optimization by Simulated Annealing : a Preliminary Computational Study for the TSP", N.I.H.E. Summer School on Combinatorial Optimisation in Dublin (July 1983), 1-9, 43-45.

*(26)* Solomon M.M., Baker E.K., Schaffer J.R., "Vehicle Routing and Scheduling Problems with Time Window Constraints : Efficient Implementations of Solution Improvement Procedures", *in* Golden B.L., Assad A.A., "Vehicle Routing : Methods and Studies", North Holland (1988), 85.

*(27)* Soriano P., "Etude des Nouvelles Avenues de Recherche proposées en Optimisation combinatoire", publication #619, Centre de recherche sur les transports, Université de Montréal (March 1989).

*(28)* Toulouse M., "La recherche avec tabous et ses applications", publication #829, Centre de recherche sur les transports, Université de Montréal (April 1992).

## SPECIAL ISSUE

# Applied Location Problems

## CALL FOR PAPERS

Just like the decision of where to set up a new facility is the first crucial question on which many other questions of logistics will critically depend, so do location problems form a field of study taking up a central place in Operations Research since its very beginnings. The simple models of the early period, such as the Weber model, well known to economists, are being extended in various ways and lots of new models allow to cope with more and more of the complexities the real world forces upon analysts and decision makers.

At the technical level location theory reflects virtually all the fields of mathematical modelling, be they linear, nonlinear, integer and/or multi-objective programming, and makes use, often at the forefront, of the most recent exact or heuristic techniques. Applications abound, not only for the traditional location of plants, distribution centres or public facilities, but also in the design of production and information systems, robotics- or pattern recognition, etc. As so often in the field of OR, it turns out that real applications are less well documented than the theory, a pity for the person in the field or the person in front of students, and that lots of new theoretical developments are not known enough to practitioners.

JORBEL plans to try taking a step towards bridging this gap by a special issue in 1996 dedicated to "Applied Location Problems". The aim is to bring together papers introducing our readers to novel trends in the field of location, either as tutorial papers and/or accounts of actual applications, which may range from short notes of a few pages of "in the field" style, or longer descriptions, possibly, but not mandatorily, including novel modelling aspects. Contributions are expected to emphasise modelling and/or presentation of applications, rather than algorithmic and/or theoretical technicalities, although some indications and/or references towards solution methods should of course be present.

Authors interested in contributing to this special issue should express their intention by sending a short abstract before September 30, 1995. The full text should be submitted in triplicate camera-ready form before January 1, 1996.

All contributions will follow the usual refereeing procedure.

Abstracts are to be sent to :

Prof. F.Plastria
INVE - Center for Industrial Location
Vrije Universiteit Brussel
Pleinlaan, 2, B-1050 Brussels, Belgium
Fax: +32 2 629.36.45, E-mail: faplastr@vub.ac.be

Full papers are to be sent to

Prof. J.Leysen
Royal Military Academy
Department of Economics and Management Science
Renaissance Ave 30, B-1040 Brussels, Belgium
Fax: +32 2 742.10.52, E-mail: leysen@rma.ac.be

All additional information and in particular prescriptions about the formal presentation of the contributions can be obtained from Prof. Leysen.