An evolutionary tabu search algorithm and the NHL scheduling problem

Daniel Costa
Département de Mathématiques
Ecole Polytechnique Fédérale de Lausanne
ORWP 92-11

May 5, 1994

Abstract

We present in this paper a new evolutionary procedure for solving general optimization problems that combines efficiently the mechanisms of genetic algorithms and tabu search. In order to explore the solution space properly interaction phases are interspersed with periods of optimization in the algorithm. An adaptation of this search principle to the National Hockey League (NHL) problem is discussed. The hybrid method developed in this paper is well suited for Open Shop Scheduling problems (OSSP). The results obtained appear to be quite satisfactory.

Keywords: Combinatorial optimization, genetic algorithms, natural evolution, tabu search, scheduling.

1 Introduction

Combinatorial optimization has received the attention of many researchers in the last two decades. A large volume of literature has been devoted to this field. This ever increasing interest is due to the fact that many practical real-world problems can be interpreted as combinatorial optimization problems (COPs) of the following form: given a set X of solutions s and an objective function f assigning to each solution s in X a real value $f(s)$, find a solution s^* in X for which $f(s)$ is minimum. Much effort has been devoted to create procedures that can lead the search towards optimal points in the solution space X. Unfortunately, the size of the solution space of a real-word problem is generally much too large to allow an exhaustive enumeration of all the points it contains.

Let us cite for example the well known traveling salesman problem (TSP) which is considered by many researchers as a "benchmark problem". Given a set of N cities and

the matrix of distances between them, the problem consists of finding a closed tour of minimum length that passes exactly once through each city. Assuming that the first city to visit is fixed and that a tour and its reverse are the same, the total number of possible tours is $(N-1)!/2$. Although computer technologies keep evolving and get more and more sophisticated, a complete enumeration of all these tours is not conceivable for large values of N. There will almost certainly exist a limit above which this problem gets intractable.

The complexity of the solution space due to the various constraints that may be inherent is another aspect that can cause difficulties when dealing with a COP. Sometimes it is really not obvious how to enumerate (even implicitly) the points in the solution space. Because of these difficulties researchers have concentrated their work on heuristics rather than on exact methods in order to solve large and complex COPs. In practice, it is generally sufficient to look for a good solution instead of an optimum which could be found only after a considerable computational effort. The challenge is to produce in a minimum time solutions as close as possible to optimal ones. Much work has been done in this field and very efficient heuristics have been developed for many of the well known COPs. However most of the methods developed are very specific to a particular problem and cannot be generalized easily to other COPs. Ideally general techniques exploiting common features in the structure of problems should be defined for approaching a wide variety of COPs. *Genetic Algorithms* (GAs), *Tabu Search* (TS) and *Simulated Annealing* (SA) are three flexible and simple algorithms which have this property. All of them have had some success in dealing with different types of COPs.

GAs are search algorithms based on genetics and biological mechanisms of natural selection for generating populations of individuals (i.e. solutions) fitter and fitter. This population evolution approach is composed of three different operators which use probabilistic rules. A general description of the various mechanisms underlying GAs is given in Section 2. Contrary to GAs, TS and SA are search procedures that deal with only one solution at a time. Each can be seen as a random step by step walk towards "good regions" in the solution space. The difference between TS and SA lies in how the step from one solution to another is selected. SA simulates the cooling of a collection of atoms by exploiting properties of Statistical Mechanics. On the other hand, TS has recourse to notions of Artificial Intelligence. It imitates human behavior by applying some learning rules to direct the search properly and to avoid undesirable loops in the random walk through the solution space. Both methods would appear to be appropriate for the scheduling problem treated in this paper. However we will concentrate on TS since it seems to be more efficient than SA. The superiority of TS over SA was shown in [2, 18] when dealing with problems related to graph coloring. We expect the same conclusion with other types of COPs. Detailed explanations about the SA procedure can be found in the technical literature [3, 20]. The basic ideas of TS are sketched in section 3.

We present in section 4 a mixed optimization method developed by Moscato [22]. In this approach periods of separate search are interspersed with interaction phases. After having discussed the advantages of this mixed strategy we consider in section 5 a highly constrained COP: the National Hockey League (NHL) scheduling problem. An adaptation of the mixed procedure to this problem, as well as some computational experiments, are reported in section 6. We show also that our procedure is well suited for solving preemptive

Open Shop Scheduling problems (OSSP) derived from the NHL problem.

2 Genetic Algorithms

Genetic algorithms define an optimization strategy derived from natural evolution rules. This evolutionary approach, developed by Holland [19] in the 70's, is applicable to a broad range of COPs. It is beyond the scope of this article to present GAs in detail. We sketch here the mechanisms of a simple GA. General explanations about GAs can be found in [7, 21].

Let us first make a short comment regarding the terminology which will subsequently be used. The objective function f is often called *fitness function* in the GAs literature. Even though this notion refers most of the time to maximization problems, we will use the adjective *fit* from time to time in this article. A fit solution (resp. population) in our terminology is a solution with a low value of f (resp. a population with a low average value of f).

Generally GAs require each solution s in X to be coded as a finite length string over some finite alphabet $\mathcal{A}$. The solution space is thus of the following form: $X = \{(a_1, a_2, \ldots, a_m) | a_i \in \mathcal{A}, |\mathcal{A}| < \infty, m < \infty\}$. The mechanism underlying standard GAs is extremely simple because it involves nothing more complex than generating random numbers, copying strings and swapping portions of strings based on some probabilistic rules. Roughly speaking the process consists of generating an initial population of individuals (i.e. strings in the solution space) and letting them evolve and interact according to well defined principles. The size of the population never changes throughout the evolution process. A GA is generally composed of three operators: Reproduction, Crossover and Mutation which are applied sequentially to the current population of individuals.

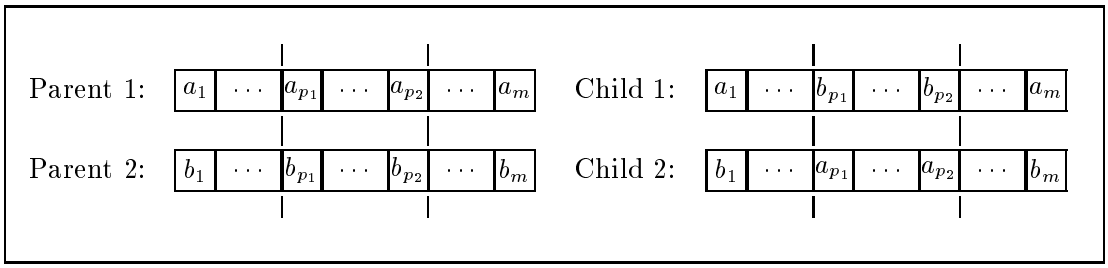
The reproduction operator is an artificial mechanism based on natural selection, the weakest individuals die off whereas the fittest proliferate. During the reproduction phase strings composing the population are simply copied according to their objective function values. The expected number of a given string in the new population will be proportional to its fitness. Since the fittest strings have a higher probability of appearing, future generations will hopefully become fitter and fitter. Once bad parent strings are eliminated, one automatically eliminates their offspring, the offspring of their offspring, and so on.

Crossover constitutes the information exchange phase of a GA which produces diversity and innovation within the population. Strings are mated randomly to give birth to new offspring. Each element thus created will have characteristics derived from both of its parents. There are different ways of performing a crossover operator. One possibility amounts to generate two positions p_1 and p_2 along the parent strings. Two new individuals can be created by swapping all the information contained between positions p_1 and p_2. This mechanism is illustrated in Figure 1.

The mutation operator can be regarded as the mechanism that generates the unavoidable accidents (or errors) occurring occasionally during a natural evolution process. It performs with a very low probability an arbitrary change in one component of a string.

Although it is relatively unimportant in comparison with the two previous operators, the role of mutation cannot be neglected. Mutation is sometimes needed to keep strings from getting stuck in local optima of the search space and to restart a process of evolution that might have stalled.

Fig 1 - Two individuals (parents) are mated to generate two new individuals (children)



Many people have doubts about the performance of such a blind algorithm. It seems indeed very strange that chance should play such a fundamental role in the search. What guarantee do we have that the populations of individuals created throughout the process will lead to solutions that are sufficiently fit? How can we expect a system to produce satisfactory results when it has so little knowledge of the problem to be solved? GA researchers claim that the answer to these questions can be found in Nature itself. The fact that Earth is now inhabited by many species which cohabit and thrive naturally in different environments is nothing else than the result of a long process of evolution having produced an enormous diversity of genetic information starting from almost nothing a few billion years ago. GAs simply mimic the mechanisms underlying this fantastic process in solving COPs. Partisans of GAs emphasize that a problem solving method does not have to be smart, but capable of learning from experience, accumulating what seems to work and rejecting what does not. Besides this biological concept underlying GAs there exists a theoretical background which has not been mentioned yet in this paper: the Schema Theorem (or the Fundamental Theorem of Genetic Algorithms). Roughly speaking this theorem says that highly fit, short-defining-length substrings (we call them building blocks) are propagated generation to generation by giving exponentially increasing samples to the observed best (see [7] for more details).

The field of GAs has kept evolving since the first developments by Holland. Descendants of GAs embody more and more an evolution that goes beyond the connotations of the term "genetic" [11]. Most of the innovations result from the introduction of problem specific knowledge into standard GAs which have shown their limits when applied to constrained optimization problems [1]. Grefenstette [14] has shown that it is possible to exploit problem specific knowledge in virtually every phase of a GA. Another trend in the evolution of GAs that goes somewhat in the same direction is the incorporation of local search heuristics into GAs. Mühlenbein *et al* [23] had some success in developing parallel GAs that allow individuals in the population to improve their fitness by local improvement.

3 Tabu search

In the same spirit as GAs, TS is a general heuristic devised for solving large COPs. However, the principles underlying the two algorithms are fundamentally different. Whereas GAs deal with a population of solutions evolving naturally, TS consists of an iterative search procedure on individual solutions. This technique was developed independently by Glover [8] and Hansen and Jaumard [16]. For a good introduction to TS, the reader is referred to [9, 10, 12, 25].

Let us define the notion of neighborhood $N(s)$ for each solution s in X. By definition $N(s)$ is the set of solutions in X reachable from s via a slight modification m. More formally, $N(s) = \{s' \in X | s' = s \oplus m, m \in M\}$ where M contains all possible modifications and '$s' = s \oplus m$' means that s' is obtained by applying modification m to s. TS starts from an initial solution randomly generated in X and moves repeatedly from a solution to a neighbor one. At each step of the procedure, a subset V^* of the neighborhood of the current solution s is generated and the local optimization problem $min\{f(x)|x \in V^* \subseteq N(s)\}$ is solved. In order to escape from local minima, the idea is to move to the best neighbor s' in V^* even if $f(s') > f(s)$. Unfortunately the algorithm is likely to cycle if moves of this type are performed without taking any precaution. In order to limit this danger, a cyclic list T is introduced. This list keeps track of the reverses of the last $|T|$ modifications which have been done during the search process. A move from s to s' will be considered as tabu if it is performed via a modification contained in T. This way of proceeding hinders the algorithm from returning to a solution reached in the last $|T|$ steps. Since only parts of the neighborhoods are explored it might be worth returning after a while to a solution visited previously to search in another direction. The concept underlying the tabu list T is unfortunately much too strict. Solutions may be considered as tabu even though they have not been encountered yet. An aspiration function A deals precisely with the rigidity of the tabu list. It permits the tabu status of a move to be dropped under certain favorable circumstances. For each value z of the objective function we define an aspiration level $A(z)$. Then a tabu move from s to s' is permitted if $f(s') < A(f(s))$ (i.e. $f(s')$ must be "small enough"). The most common example of application of this principle is obtained in setting $A(z) = f(\tilde{s})$ where $\tilde{s}$ is the best solution reached from any solution s such that $f(s) = z$. Initially $A(z) = z$ for all possible values of $z = f(s)$. This aspiration function is updated as follows whenever we move from s to s':

$$A(f(s)) = min(A(f(s)), f(s'))$$
$$A(f(s')) = min(A(f(s')), f(s))$$

The above shows that the reverse move from s' to s is considered in the updating of A even though it was not done explicitly.

Generally the whole process is stopped as soon as a given number of iterations have been performed without improving the best solution obtained. Figure 2 outlines the general TS procedure.

Figure 2 - The general tabu search

Initialization

generate a random solution s in X ;
$s^* := s$; (best solution reached so far)
$iter := 0$; (iteration counter)
$bestiter := 0$ (iteration at which the best iteration has been found)
$T := \emptyset$;
initialize the aspiration function A ;

While $(iter - bestiter < niter\_max)$ **do**
$iter := iter + 1$;
generate a set V^* of solutions $s_i = s \oplus m_i$ such that either $m_i \notin T$
or $f(s_i) < A(f(s))$;
choose the best solution s' in V^* ;
update the tabu list T and the aspiration function A ;
If $(f(s') < f(s^*))$ **then**
$s^* := s'$;
$bestiter := iter$;
$s := s'$;

---

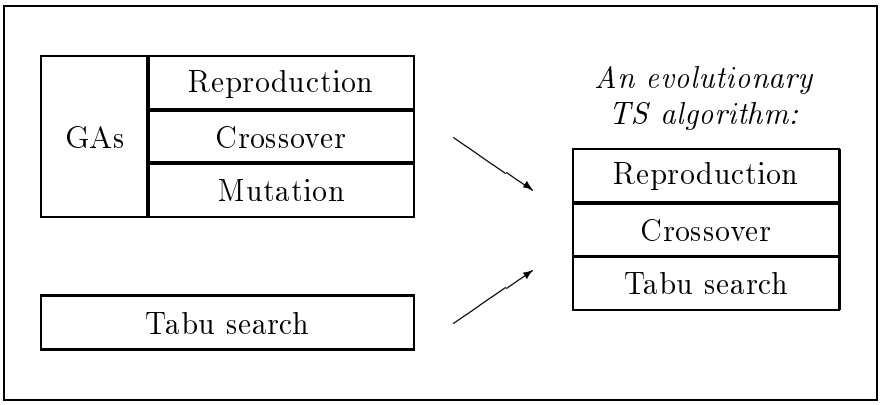
4 Tabu search combined with genetic algorithms

Tabu search is one of the most efficient algorithms for solving COPs among the various available search procedures. The principle of searching repeatedly for a neighbor as good as possible seems to lead towards interesting areas in the search space quickly and with a high probability. It should be pointed out that contrary to SA there exists no convergence theorem for TS. TS has several desirable qualities: it is powerful, very flexible and easy to implement. Unfortunately there is one aspect which is troublesome: various parameters have to be adjusted in the algorithm. From this point of view, TS is not very robust. The parameters have a direct influence on the quality of the final solution. A good knowledge of the problem is required to fix them appropriately. At this level, GAs are to some extent complementary to TS. There are few parameters to adjust and they deal with a population of solutions rather than with a single solution.

The question which now arises naturally is the following: *"Can TS and GAs be combined efficiently to create a new hybrid strategy taking advantage of their complementary features ?"* The answer is *yes*. Such a concept was developed recently by Moscato [22] for tackling some test problems. His model is based on a set of individuals, called *agents*, which are arranged topologically around a 'ring'. Periods of competition, cooperation

and individual optimization intersperse throughout the process according to well defined rules. A similar approach, closer in a certain sense to GAs, is investigated in this paper. Notions of Artificial Intelligence are introduced to guide the natural evolution process. The mutation phase of GAs is replaced by a search in the solution space governed by a tabu algorithm. Instead of random mutation changes in its components, each individual undergoes a separate optimization process before interacting again with other members of the population. The phases of such an evolutionary TS algorithm (ETS) are schematized in Figure 3.

Figure 3 - TS combined with GAs



We expect the solutions produced by such a mixed algorithm to be better than those obtained without any interaction. The combination of TS with a regular collective exchange of information is likely to yield a search procedure much more efficient and robust than a classical TS.

A very highly constrained problem is presented in the next section. The combined method presented in this section will be used to solve it.

5 The NHL game scheduling problem

Constructing a schedule for a sport league is a very difficult task. There are generally a large number of entities involved in the process and hence many requirements to take into account. The NHL game scheduling problem is certainly a good illustration of this point. Because of the various types of constraints involved, it is perhaps one of the most complex applications of its kind.

5.1 General description

The National Hockey League includes teams located in North America and is divided into two Conferences, each of which is split into two Divisions. The alignement of teams for the 1993-94 NHL season is shown in Table 1.

Table 1 - Teams in the League

Western Conference		Eastern Conference	
Pacific Division	Central Division	Northeast Division	Atlantic Division
Anaheim	Chicago	Boston	New Jersey
Calgary	Dallas	Buffalo	NY Islanders
Edmonton	Detroit	Hartford	NY Rangers
Los Angeles	St. Louis	Montreal	Philadelphia
San Jose	Toronto	Ottawa	South Florida
Vancouver	Winnipeg	Pittsburgh	Tampa Bay
		Quebec	Washington

In this paper we deal with the 1989-90, 1991-92 and 1993-94 situations. Unless specified, we use the present tense for the 1993-94 season and eventual comments for the two earlier seasons are put in brackets. In recent years, the NHL expanded from 21 to 26 teams (1989-90: 21 teams/1991-92: 22 teams) and it is expected that up to 28 teams may be part of the NHL by the year 2000. A total of 1066 (800/880) regular games are played during the season. Each team plays 41 (40/40) games at home (H) and 41 (40/40) away (A). Teams in the Pacific and Central Divisions play other teams within their own division 6 times (3H,3A). In the Northeast and Atlantic Divisions teams play other teams within their own division only 5 times (some 3H,2A and others 2H,3A). All teams play 4 games (2H,2A) against each of the teams in the other Division of their Conference and 2 games (1H,1A) against each of the teams in the other Conference. A procedure based on integer linear programming to generate game allocation scenarios is proposed in [5]. The season begins in early October and ends in early April (26-28 weeks). In 1992-93, the NHL and the NHL Players' Association decided to play extra games in neutral sites for promotional purposes. For the second successive year, teams play two games in non-NHL cities during the 1993-94 season.

The NHL problem consists of assigning each game a date while taking the following constraints into account:

1) A team cannot play more than one game a day.

2) Arena availabilities are handled essentially by the teams. Each team manager provides a set of 56 (50/50) or slightly more preferred home dates from which 41 (40/40) are to be chosen by the scheduler.

3) The total distance travelled by the teams should be minimized.

4) A team should not play games on three straight days and should not play more than three games in five consecutive days.

5) A team should play regularly while it is not at home. Breaks of more than 3 days should be avoided on the road.

6) Games cannot be scheduled on certain days, either because of a major event (All Star Game break) or a holiday (Christmas Eve, Christmas) or the unavailability of a team (e.g. a special event in which the team is involved).

7) Some arenas cannot be used on certain days because of other major activities (circus, music concerts, sporting events, ...).

8) There should be an even distribution of games throughout the season. Identical games (i.e. the same pair of teams at the same location) involving teams of the same division should be scheduled at least 14 days apart. When teams come from different divisions the minimum allowed lapse of time between revisits is 30 days.

9) A team cannot play more than 7 consecutive road games and the duration of a road trip cannot exceed 14 days.

10) One idle day must be provided between games involving lengthy travel. Two games cannot be scheduled on two consecutive days if one of the teams involved has to travel more than 900 miles. By convention this applies only to teams coming from different divisions.

Minor restrictions and constraints specific only to certain teams have been purposely omitted in the above description to make the presentation of the problem clearer. Also some points dealing with the flexibility of the team managers have not been considered in our approach because they are very difficult to incorporate in a computer program.

As we can expect the NHL scheduling problem is far from easy to solve, even in the slightly simplified form presented above. The problem looks like a highly constrained multi-person TSP where each salesperson has to visit each city a given number of times. Despite (or perhaps because of) its complexity the schedule continues to be developed on a manual basis. Once a first draft is prepared, the central scheduler gets in touch with the different team managers for possible suggestions and modifications. This negotiation phase deals essentially with the arena availability which is undoubtedly the most difficult type of constraint to satisfy. The 56 (50/50) possible home dates proposed by the team managers are generally not enough to produce a feasible schedule. Some changes are performed by the team managers to allow the central scheduler to find a final schedule satisfying the requirements of all the clubs in the League.

To our knowledge the NHL scheduling problem has already been tackled twice in the literature. W. Fraser [6] developed a 'road trip simulator' module which has proven to be rather inflexible for properly scheduling all games. His conclusion was: *"Due to the unpredictability of some constraints and the human relationships involved in others, it is impractical to expect that a computer program will ever produce a final schedule which would require no tuning or adjustment"*. J. Ferland and C. Fleurent [4] on a second attempt proposed a decision support system for building the schedule in an interactive manner. Their approach involves at the same time the experience of the program user and sophisticated computerized procedures.

5.2 Mathematical formulation

The NHL problem presented above can be formulated as a combinatorial optimization problem in the following fashion.

Let us split into two categories C_a and C_b the set C of all the constraints described in the previous section. Constraints in C_a are called *essential* while those in C_b are termed *relaxed*. A schedule is *acceptable* if it satisfies all the constraints in C. If it satisfies at least the essential constraints it is said to be *feasible*.

The following objective function (or fitness function) evaluates the unacceptability of a feasible schedule S:

$$f(S) = \sum_{i \in C_b} w_i \cdot f_i(S)$$

$f(S)$ is defined through a set of weights w_i giving relative importance to each relaxed constraint. $f_i(S)$ computes the degree of violation of the i-th relaxed constraint in the schedule S. The problem amounts to looking for a schedule S^* that minimizes the value of f over the set X of feasible schedules. The size and the structure of X, as well as the complexity of f, result directly from the way of partitioning the set C. Some criteria are presented in [17]. A game should never be scheduled on a day if such an assignement induces nothing but an unacceptable schedule. On the other hand a conflict between two games is not a sufficient condition for hindering an assignement. Based on these principles, it was decided to include constraints 6 to 8 in C_a and to relax constraints 4 and 5. Constraint 1 should be relaxed but this has to be done with care if we want to avoid manipulating messy schedules. In particular the notions of road trips and home games need to be differentiated properly. Thus we split constraint 1 into three parts as follows:

(1.1) A team cannot play more than one game per day.

(1.2) A team cannot play more than two games per day.

(1.3) A team cannot play at home and on the road during the same day.

In order to have feasible schedules relatively easy to handle, we relaxed (1.1) and put (1.2) and (1.3) in C_a. Constraint 2 has also been relaxed because it is hard (probably impossible) to produce a schedule satisfying this constraint accurately. The relaxation of constraint 3 is natural since we are interested in finding a schedule that minimizes the distance travelled by the teams. In order to reduce the number of components f_i constraints 9 and 10 have been included in C_a. Their influence on the structure of X is negligible. The role of each component $f_i(S)$ is described below:

- $f_1(S)$ indicates how many times a team plays two games per day.
- $f_2(S)$ indicates how many times the unavailability of an arena is not respected.
- $f_3(S)$ indicates the total distance (in hundred of miles) travelled by the teams during a whole season.

- $f_4(S)$ indicates how many times a team plays more than two games on three straight days or more than three games on five straight days.
- $f_5(S)$ indicates how many breaks of more than three days occur while a team is on the road.

We discuss in the remainder of this paper an adaptation of the mixed procedure of section 4 to the NHL problem. Despite W. Fraser's pessimism we expect our algorithm to produce implementable results, even though final minor manual tunings may always be necessary to satisfy all the League requirements.

6 An evolutionary TS for the NHL problem

The ETS algorithm introduced in this paper is made up of 3 phases which succeed repeatedly throughout the process. After having explained how to generate an initial population of schedules, we present the mechanisms underlying each phase of the algorithm used for building the NHL schedule. Various computational experiments are reported at the end of this section.

6.1 Generation of an initial population of schedules

The method we adopted to generate an initial schedule is somewhat similar to the one presented in [4]. Road trips are built sequentially for each team in the League while meeting every essential constraint. The road trip generation procedure which is sketched in Figure 4 is based on the following notation:

(1) Denote by $R(t)$ the set of games remaining to be scheduled on the road for team t. Initially $|R(t)| = 41(40/40)$ for every team t. Let $T = (t_1, \ldots, t_n)$ be the list of teams sorted by decreasing order of $|R(t)|$ $(i < j \Rightarrow |R(t_i)| \geq |R(t_j)|)$.

(2) Consider for each team t the set of intervals (d_1, d_2) where d_1 and d_2 are two consecutive available home dates provided by t such that $|d_2 - d_1| \geq 3$. These intervals are called *free intervals*. In principle a team plays only games on the road during a free interval. The minimum duration of a free interval is set at 3. Let $I(t)$ be the list of free intervals for team t sorted by length in decreasing order.

(3) $[t_1, t_2]$ refers to a game involving teams t_1 and t_2 that has to be played in the arena of team t_2.

A whole population $\mathcal{P}$ of schedules can be obtained with the technique presented in Figure 4 by rearranging the initial list T and ordering somewhat differently the various lists of free intervals $I(t)$. We give below the average values of the components f_i characterizing a schedule S (for the 1991-92 season) produced by the road trip generator of Figure 4:

$$f_1 \simeq 230; f_2 \simeq 60; f_3 \simeq 8200; f_4 \simeq 1100; f_5 = 0;$$

Constraint 5 happens to be satisfied in every initial schedule contrary to the other relaxed constraints. The value of f_4 is relatively high because of the $f_1 \simeq 230$ teams playing two games on the same day in the schedule. More effort could be put into establishing a fitter initial population of schedules. This was not done since there is no guarantee that this will eventually lead to better results. We observed that the initial population does not significantly influence the quality of the final generation of solutions. As the availabilities of the arenas in neutral sites are not known, neutral games in the 1993-94 season are scheduled according to the official schedule established by the League. ETS is not allowed to move these games.

Figure 4 - Generation of an initial schedule

$ngames\_to\_schedule$:= 1066 (800/880);
$ngames[t_1, t_2]$:= number of games $[t_1, t_2]$ remaining to be scheduled;

While $(ngames\_to\_schedule > 0)$ **do**
- t_1 := first team in the list T;
- Pick the first free interval (d_1, d_2) in $I(t_1)$ containing a feasible and available home date d for a team t_2 such that $ngames[t_1, t_2] > 0$. If no interval can be found in this fashion, then the requirement of arena availability on day d is dropped;
- If team t_2 is not unique, pick the one which is located the furthest away from t_1;
- If day d is not unique pick the one which is the closest to the middle of the interval (d_1, d_2);
- Schedule game $[t_1, t_2]$ on day d ;
- Complete the trip by scheduling games $[t_1, t]$ in the intervals $I_1 = (d_1, d)$ and $I_2 = (d, d_2)$ alternatively while meeting the essential constraints. The arena availability constraint is taken into account whenever it is possible. Days in I_1 (resp. in I_2) are considered by decreasing (resp. increasing) order starting from d. When scheduling a new game $[t_1, t]$ in I_1 (resp. in I_2) priority is given to teams t located close to the last team t_{I_1} (resp. t_{I_2}) scheduled in I_1 (resp. I_2). Initially $t_{I_1} = t_{I_2} = t_2$.
- Update $T, I(t_1), ngames\_to\_schedule$ and $ngames[t_1, \cdot]$;

6.2 The reproduction phase

When solving a maximization problem, the easiest way to implement a reproduction operator is to associate to each schedule S a probability $p(S)$ equal to its fitness $f(S)$ divided by the sum of the fitnesses of all the members of the population. In our case the

probabilities $p(S)$ need to be defined differently. We need a function such that $p(S)$ is monotonically decreasing with $f(S)$. Let $M = f_{max} + (f_{max} - f_{min})/n$ where f_{max} (resp. f_{min}) denotes the fitness of the worst (resp. the best) schedule in the current population $\mathcal{P}$ of size n. A new population $\mathcal{P}$ is created by copying a schedule S according to the following reproduction probability:

$$p(S) = \frac{M - f(S)}{n \cdot M - \sum_{\tilde{S} \in \mathcal{P}} f(\tilde{S})}$$

The most popular way of implementing this competition process is to create a biased roulette wheel where each schedule has a roulette wheel slot sized in proportion to $p(S)$. A simple spin of the wheel yields a reproduction candidate. The repoduction phases consists then of n spins of the wheel. In the new population thus created the expected number of offspring of a given schedule S is clearly equal to $n \cdot p(S)$. The constant M has been chosen in order to give a positive but relatively small reproduction probability to the least fit schedule in $\mathcal{P}$.

6.3 The crossover phase

The definition of an efficient crossover operator is the crucial point when implementing a natural evolution process for solving optimization problems. A good knowledge of the problem under study is required. Various attempts were made before finding an operator that properly guides the cooperation phase in the NHL problem context.

Exchange of information occurs between two schedules on the basis of a *cooperation request* of one schedule to the other. A new schedule is created every time a schedule S_a addresses a cooperation request to a schedule S_b. Let us denote such a request by $S_a \rightarrow S_b$. In order to make things more explicit when describing the crossover operator underlying a cooperation request let us introduce some definitions. σ denotes a permutation of the elements $1, 2, \ldots, n$. $ndays$ is the number of days in a regular season. The *frame* $F_{S,t}$ of the schedule of a team t in S is a string of length $ndays$ with 0-1 components. $F_{S,t}[d] = 0$ if team t is at home on day d in the schedule S. $F_{S,t}[d] = 1$ whenever team t is on the road. By convention, a team is at home between the last game on the road (resp. at home) and the next game at home (resp. on the road). Also a team is assumed to be at home before its first game and after its last game of the season. Without going too much into detail, let us say that the *cost* of a game g in a schedule is a measure based on the degree of violation of the relaxed constraints due to game g. Finally the word *redundant* will be used in the following sense. For a given visitor-home team pair, if the number of such games scheduled is greater than the number specified by the league, then all of the games scheduled are called redundant.

Before every cooperation phase, a random permutation σ is used to rearrange schedules $S_1, S_2, \ldots, S_n$ within the population. Then every schedule $S_{\sigma(i)}$ $(i = 1, 2, \ldots, n)$ sends a cooperation request to $S_{\sigma(i \bmod n+1)}$ $(S_{\sigma(i)} \rightarrow S_{\sigma(i \bmod n+1)})$. Roughly speaking this consists of putting together within a single schedule S_i^* as many games as possible without

modifying the frame of schedule $S_{\sigma(i \bmod n+1)}$. A feasible schedule is obtained by sequentially removing the most costly redundant games in S_i^*. In order not to lose the frame of the best schedule involved in a cooperation request, a request $S_{\sigma(i)} \rightarrow S_{\sigma(i \bmod n+1)}$ is accepted under certain circumstances only. If $S_{\sigma(i)}$ is fitter than $S_{\sigma(i \bmod n+1)}$ and if a request $S_{\sigma(i \bmod n+1)} \rightarrow S_{\sigma(i)}$ has not been performed yet in the current cooperation phase (as a reminder, a schedule may appear more than once in the population) then the reverse request $S_{\sigma(i \bmod n+1)} \rightarrow S_{\sigma(i)}$ is used instead.

The detailed mechanism underlying a cooperation request $S_a \rightarrow S_b$ is presented in Figure 5. The schedules S_i^* thus created make up the next population of individuals which will go through the next stage of the ETS algorithm.

Figure 5 - Creation of a schedule S^* via a cooperation request $S_a \rightarrow S_b$

- Make a copy of S_b and call it S^*
- $F_{S^*,t} = F_{S_b,t}$ for every team t
- For every day d and for every game $[t_1, t_2]$ played on day d in S_a, insert $[t_1, t_2]$ on day d in S^* if the corresponding frames are compatible (i.e if $F_{S_a,t_1}[d] = F_{S^*,t_1}[d]$ and $F_{S_a,t_2}[d] = F_{S^*,t_2}[d]$)
- Establish a list $\mathcal{L}$ of all the redundant games in S^*
- Sort the list $\mathcal{L}$ by cost in decreasing order

While $(\mathcal{L} \neq \emptyset)$ **do**

- Remove from S^* the first game g in $\mathcal{L}$
- Remove game g from $\mathcal{L}$
- If games identical to g (same visitor versus same home team) are no longer redundant in S^* then remove them from $\mathcal{L}$
- Resort the list $\mathcal{L}$ if any remaining costs have changed

---

6.4 Tabu Search

We present in this section an adaptation of the general TS procedure to the NHL problem.

The neighborhood $N(S)$ of a schedule S consists of all the schedules that can be obtained from S by moving a single game from one day to another. In order to more quickly reach good regions of the set of feasible schedules, only games violating at least one relaxed constraint (except for the distance constraint whose violation is hard to evaluate for a single game) are moved around during the TS process. These games are called *conflicting games*. A maximum number *nneigh* of pairs (g_i, w_i) are generated at each step of the search. g_i is a conflicting game and w_i is a week different from the one in which game g_i is currently scheduled (we do not consider moves within the same week because the game was probably moved earlier into its current position at which time the best date

within the week was selected). For every game g_i, we look for the best feasible day d_i in week w_i. The best of the moves $(g_i \rightarrow d_i)$ thus created will be retained for building the next schedule. In order to reduce the time spent for choosing a neighbor, we move directly to the best neighbor found after having generated at least $\lfloor nneigh/2 \rfloor$ neighbors if it happens to be better than the current solution S. The number of neighbors to enumerate at each step should depend on the size of $N(S)$. It was decided to set $nneigh = prop_N \cdot |N(S)|$ where $0 < prop_N < 1$.

In order to prevent cycling, the reverses of the last moves are memorized in the tabu list T. If a game g is moved from day d^1 to day d^2 at a given step of the algorithm then the move $(g \rightarrow d^1)$ is considered as tabu for the next $|T|$ iterations. To diversify a bit the exploration in the search space once in a while, the size of the tabu list is randomly generated between two given bounds t_{min} and t_{max} [2]. The aspiration criterion presented in section 3 has been slightly modified in this adaptation of TS. Instead of considering the function $f(S) = \sum_{i \in C_b} w_i \cdot f_i(S)$ when implementing $A(z)$ we have concentrated on the first two relaxed constraints and have used the function $g(S) = f_1(S) + f_2(S)$. The weights w_1, w_2 and the components f_3, f_4, f_5 have been omitted in the definition of g to reduce significantly the number of values $A(z = g(S))$ that we have to keep track of throughout the TS process. Finally it was decided to stop every TS procedure after a given number $niter$ of iterations independently of the iteration at which the best solution was found.

A population of individuals is likely to converge prematurely if either $niter$ or n are small. We say that a population goes through a diversity crisis when it contains more than $\lfloor n/3 \rfloor$ copies of a same solution S after a cooperation phase. For half of these copies, if the fitness of S has not been increased in the next individual optimization phase after $\lfloor niter/2 \rfloor$ TS iterations then the solution we retain at the end of each of the searches considered is the best solution (different from S) obtained in the last $\lfloor niter/2 \rfloor$ TS iterations. Hopefully the diversity incorporated in this way will propagate in the subsequent generations giving birth to fitter solutions.

6.5 Numerical results for the NHL problem

The experiments presented here have been performed on a Silicon Graphics workstation (9 Mflops). In order to select appropriate values of the parameters governing the search various trials have been carried out with the computer code using the data of the 1989-90 and 1991-92 NHL seasons. Let us call TS_n an algorithm composed of n classical TS procedures running separately (i.e. without any interaction between two processes). Similarly ETS dealing with a population of size n is denoted ETS_n.

In the definition of the objective function we chose $w_1 = 26, w_2 = 13, w_3 = 1, w_4 = 10$ and $w_5 = 2$. These weights give solutions which are comparable to the schedules obtained manually by the NHL schedulers. Increasing (resp. decreasing) a weight w_i would give more (resp. less) importance to the associated relaxed constraint.

To find appropriate values for the tabu list size $|T|$, we investigate the behaviour of TS_5 with $|T|$ ranging from 0 to 300. Figures 6 and 7 show the average results produced after

$niter = 5,000$ iterations for $prop_N = 0.05$, $prop_N = 0.2$ and $prop_N = 0.4$. We observe that the best solutions are obtained when $|T|$ lies in [10,80]. In a more general way $|T|$ should be chosen proportionally to the number of possible moves (i.e. $ngames \cdot nweeks$) in the search space. Since this number does not change much in the 3 NHL problem instances considered in this paper it has been decided to generate T between $t_{min} = 10$ and $t_{max} = 80$ once every 50 iterations independently of $ngames$ and $nweeks$. Whereas $prop_N = 0.2$ and $prop_N = 0.4$ yield similar results, we notice that there is a significantly loss of quality when considering $prop_N = 0.05$. Other tests have shown that generating at most a fifth (i.e. $prop_N = 0.2$) of the conflicting games is a good compromise between elapsed CPU time and average quality of the solutions obtained.

Figure 6 & Figure 7

Let $ngen$ be the number of generations created in ETS_n (each generation goes through a reproduction phase, a cooperation phase and a TS phase) and $nitertot$ ($= ngen \times niter$) the total number of TS steps performed by a member of the population. The parameters $ngen$ and $nitot$ need now to be tuned in such a way that $ngen \times niter = nitertot$. The larger $nitertot$ is, the higher the quality of the results obtained is likely to be. We have decided to set $nitertot = 20,000$. Experiments with 5 different initial populations of size 8 ($5 \times \text{ETS}_8$) have been conducted. The results obtained with $ngen$ ranging from 1 to 80 are shown in Table 2. A number of generations smaller than 4 is clearly insufficient. On the other side, considering a large number of generations does not allow TS to seek efficiently in a given direction. The best results are obtained with decompositions 8×2500, 10×2000, 20×1000 and 40×500. We will use decomposition 20×1000 in the remaining algorithms.

Table 2 - Number of generations versus number of TS iterations

$ngen \times niter$	1989-90	1991-92
1×20000	9,003.82	9,509.27
2×10000	8,911.17	9,465.02
4×5000	8,779.33	9,331.40
5×4000	8,784.75	9,346.97
8×2500	8,746.27	9,258.32
10×2000	8,699.42	9,273.80
20×1000	8,708.07	9,254.57
40×500	8,711.72	9,313.65
50×400	8,832.62	9,312.80
80×250	8,851.97	9,359.27

The last parameter to study is the size n of the population. To that purpose we run ETS_n 5 times with n varying from 1 to 50. As it can be seen in Table 3, the quality of the final solutions increases when n goes from 1 to 8. For larger values of n this progression is less pronounced. We think that the size of the population does not have a significant influence on the results for values of n larger than 30-40. Because of the high CPU time

required, no experiment has been carried out with $n > 50$ (it takes about 10 minutes to run $20,000$ TS iterations on a single individual).

Table 3 - Influence of the size n of the population

n	1989-90	1991-92
1	8,923.60	9,457.40
2	8,799.60	9,390.40
3	8,771.79	9,336.53
5	8,728.40	9,166.64
8	8,619.00	9,155.10
10	8,626.64	9,158.40
20	8,534.84	9,072.70
30	8,515.79	8,952.81
40	8,441.30	9,049.11
50	8,482.62	8,882.83

In order to measure the performance of ETS we compare in Table 2 the results produced by ETS_8 and TS_8 with the data of the 1989-90, 1991-92 and 1993-94 NHL seasons. For each of these 3 instances, ETS_8 and TS_8 are given the same initial population of schedules. The first three columns of Table 4 report respectively the average fitness, the fitness of the worst schedule and the fitness of the best schedule in the final population obtained after $ngen = 20$ generations ($nitertot = 20,000; niter = 1,000$). The last five columns give the values of the different components f_i characterizing the fittest schedule in the population.

Table 4 - NHL results

season	method	f_{av}	f_{max}	f_{min}	f_1	f_2	f_3	f_4	f_5
1989-90	ETS	8,531.00	8,556	8,513	0	91	7,262	3	19
	TS_8	8,970.25	9,059	8,827	0	91	7,550	4	27
	Official Schedule	-	-	10,210	0	148	8,176	4	35
1991-92	ETS	9,055.00	9,111	9,032	0	80	7,916	3	23
	TS_8	9,519.63	9,704	9,437	0	95	8,114	4	24
	Official Schedule	-	-	10,963	0	150	8,889	3	47
1993-94	ETS	12,384.75	12,392	12,347	0	105	10,896	2	33
	TS_8	13,028.63	13,284	12,863	0	124	11,091	7	45
	Official Schedule	-	-	13,293	0	111	11,718	3	51

We observe that the principle of inserting cooperation and competition phases during a TS process increases significantly the performance of the algorithm. The various experiments we have carried out on the NHL problem show us that the final population of schedules produced by ETS_8 is on average between 3 and 6 % fitter than the one obtained by TS_8. As expected, ETS_8 provides a population of schedules more homogeneous than TS_8 does. The range of the values f in the final population is equal to 45 (43/79) for ETS_8

and to 421 (232/267) for TS_8. The higher variability of the final population achieved by TS_8 is due to the fact that there is no exchange of information between the schedules throughout the algorithm. Thus, the influence of the initial population on the final one is more important in TS_8 than in ETS_8.

The characteristics of the official NHL schedules are also reported on Table 5. It is important to point out that 4 out of the 5 essential constraints are not satisfied thoroughly in these schedules. According to the data we obtained, 1 (6/6) game(s) is (are) scheduled on a day when the home team arena is supposed to be busy, 0 (0/2) team(s) has (have) to play while it is (they are) supposedly unavailable, minimum spacing between identical games is violated 25 (44/33) times and 12 (6/3) teams are involved in two consecutive games requiring a long journey (> 900 miles). As mentioned earlier, a NHL schedule is obtained after lengthy discussions involving the central scheduler and the team managers. Various concessions are made by both sides in this negotiation phase. This explains partially why the original arena availability constraint and most of the essential constraints are not well respected in the official schedule. From a fitness function point of view the schedules produced by ETS_8 are much better than the official ones. It turns out that we managed to reduce by 82,200 (91,400/97,300) miles the total distance travelled during a regular season. Because of the contradictory objectives expressed by the arena availability constraint and the distance constraint, the total distance travelled by the teams in the League can be reduced even more by decreasing w_2 and increasing w_3 somewhat. This depends on which constraint has higher priority.

Unfortunately, it is impossible to evaluate precisely the gap existing between the objective expressed by f and the real goals of the team managers. The central scheduler could use the calendar produced by ETS as a first draft before beginning the unavoidable interaction phase with the team managers. Despite this, the ETS schedules are quite reasonable in our opinion.

The running time of an ETS algorithm is negligible in comparison with the amount of time spent by the central scheduler in developing a schedule. Twenty generations of 8 schedules have been achieved in approximately 80 minutes of CPU time on a sequential computer. Due to its asynchronicity and intrinsic parallel nature, ETS is well suited for parallel computation. The use of a parallel MIMD machine would have significantly reduced the execution time of the algorithm. Investigations to evaluate the speed-up of the algorithm have not been carried out in this study.

6.6 A variant of the NHL problem

The NHL problem can be seen as a generalization of two problems commonly related in the technical literature: the multi-person traveling salesman problem (m-TSP) [15] and the open shop scheduling problem (OSSP) [13]. In order to demonstrate the effectiveness of ETS method in a more general way we generate here a variant of the NHL problem that can be seen as a multiple-constraint preemptive OSSP. Let us split the teams in the league into two sets $\mathcal{M} = (M_1, M_2, \cdots, M_r)$ and $\mathcal{J} = (J_1, J_2, \cdots, J_s)$. Teams in $\mathcal{M}$ play only on the road whereas teams in $\mathcal{J}$ stay always at home. In the open shop context,

machines and jobs identify with teams in $\mathcal{M}$ and $\mathcal{J}$ respectively. Each job J_j consists of tasks $T_{1j}, T_{2j}, \cdots, T_{rj}$ which have to be processed on $M_1, M_2, \cdots, M_r$ respectively. The order of the processing is not fixed. The processing time p_{ij} (given in periods) of each task T_{ij} is an integer randomly generated between 1 and 10. Preemptions are allowed, i.e. a task T_{ij} in process on M_i can be interrupted and continued later. All tasks have to be processed within $nperiods = 2 \cdot \max(\max_{i \in \mathcal{M}}(\sum_{j \in \mathcal{J}} p_{ij}), \max_{j \in \mathcal{J}}(\sum_{i \in \mathcal{M}} p_{ij})$. The distance d_{kl} (in hundred of miles) between team k and team l ($k, l \in \mathcal{J}$) refers now to the set up cost for a machine M_i to process T_{il} after T_{ik} (cf [24] for an application of such a model). Every machine needs a maintenance service once in a while. The cost of a service on M_i is equal to $d_{ki} + d_{il}$ where T_{ik} is the last task processed on M_i before the service and T_{il} the task processed just after the service. Among the constraints mentioned in section 5.1 only the four sketched below are taken into account here :

1) No two tasks of the same job can be processed simultaneously and each machine works on at most one task at a time.

2) Because of external availability and/or delivery constraints (e.g. release dates, and due dates) the tasks of every job should be processed on some specified periods. A set of $m_j = \lfloor \frac{3}{2} \cdot (\sum_{i \in \mathcal{M}} p_{ij}) \rfloor$ randomly generated periods is provided for each job J_j.

3) The sum of set up costs and service costs over all machines should be minimized.

9) A machine cannot process tasks for more than 7 periods without any maintenance service. It follows that the number of tasks processed between two services is bounded by 7.

Except for the generation of an initial solution where ficticious unavailable periods for the machines are considered, the method presented in the previous sections can be used for the above OSSP without modification. As mentioned earlier the size of the tabu list should be tuned according to the number of possible moves in the search space. Based on the results presented in Figures 6 & 7 and on the average ratios $t_{min}/ngames * nweeks$ and $t_{max}/ngames * nweeks$ obtained when dealing with the 1989-90 and 1991-92 NHL problems we set $t_{min} = \lfloor 0.0005 * ntasks * nper_7 \rfloor$ and $t_{max} = \lfloor 0.0036 * ntasks * nper_7 \rfloor$ where $ntasks = \sum_{i \in \mathcal{M}} \sum_{j \in \mathcal{J}} p_{ij}$ is the total number of tasks and $nper_7 = \lfloor nperiods/7 \rfloor$ the number of 7 period intervals. The 22 teams in the NHL during the 1991-92 season and the matrix of distances associated are taken into account in this application ($|\mathcal{M}| + |\mathcal{J}| = 22$). The parameter $nitertot$ is reduced from 20,000 to 5,000 and the decomposition ($ngen \times niter$) is chosen proportionally to the one retained for the NHL problem, i.e. $ngen = 10$ and $niter = 500$. The weights w_1, w_2, w_3 are kept unchanged ($w_4 = w_5 = 0$). Table 5 shows the average results produced by ETS_8 and TS_8 when considering a set of 4, 8 and 12 machines. For a given number of machines, ETS_8 and TS_8 are run 10 times, each time with a new set of machines M_i and different processing times p_{ij}. All the schedules we obtain are overlap free (i.e. $f_1 = 0$). $\bar{f}_{min}$ (resp. $\bar{f}_{max}$) reports the average fitness of the best (resp. worst) solution in each of the 10 final populations achieved. $\bar{f}$ is the average value of f over all the solutions obtained. The ranges of f, f_2 and f_3 are also indicated in Table 5.

We notice once again that ETS_8 performs significantly better than TS_8. The conclusions achieved in the previous subsection are the same here, i.e. ETS_8 gives birth to a final population of schedules which is fitter and more homogeneous than TS_8.

Table 5 - Results obtained by ETS_8 and TS_8 on the OSSP

r	obj. function	ETS_8	TS_8
4	$\bar{f}_{min}$	3,181.10	3,443.60
	$\bar{f}_{max}$	3,226.10	3,731.60
	$\bar{f}$	3,204.55	3,589.90
	f	2,620 $\rightarrow$ 3,604	3,159 $\rightarrow$ 4,002
	f_2	4 $\rightarrow$ 34	8 $\rightarrow$ 31
	f_3	2,438 $\rightarrow$ 3,279	2,951 $\rightarrow$ 3,781
8	$\bar{f}_{min}$	3,338.30	3,446.00
	$\bar{f}_{max}$	3,382.40	3,679.40
	$\bar{f}$	3,359.59	3,562.36
	f	2,826 $\rightarrow$ 3,670	2,920 $\rightarrow$ 3,939
	f_2	0 $\rightarrow$ 7	0 $\rightarrow$ 7
	f_3	2,757 $\rightarrow$ 3,639	2,881 $\rightarrow$ 3,913
12	$\bar{f}_{min}$	3,125.40	3,268.20
	$\bar{f}_{max}$	3,167.60	3,481.20
	$\bar{f}$	3,151.75	3,364.01
	f	2,794 $\rightarrow$ 3,654	2,859 $\rightarrow$ 3,971
	f_2	0 $\rightarrow$ 2	0 $\rightarrow$ 2
	f_3	2,794 $\rightarrow$ 3,654	2,859 $\rightarrow$ 3,971

7 Conclusion

The main purpose of this paper was to introduce an evolutionary approach capable of tackling a wide variety of problems. We asked ourselves whether the TS method might be improved by modelling some aspects of biological optimization strategies. After having sketched the basic mechanisms underlying GAs and TS, we have shown that the principle of mixing these two fundamentally different search procedures leads to interesting optimization properties. The blend of the advantages of GAs and TS yields a new evolutionary procedure which is expected to enhance the performance of both algorithms running separately. The originality of an ETS algorithm lies in the succession of interaction phases (competition + cooperation) and individual search phases throughout the process.

We have seen that ETS performs significantly better than TS when dealing with two highly constrained scheduling problems. The results we have obtained for the NHL problem would appear to be quite promising. All the constraints we have taken into account happen to be satisfied in the ETS calendars better than in the official ones established manually by the NHL. Even though it is hard to evaluate the real objectives of the various people involved in the process of establishing the schedule we expect the

calendar produced by an ETS algorithm to be quite usable. Some manual adjustments may be necessary but this is a small task in comparison with the total amount of time currently spent by the League in establishing a schedule every year.

As mentioned earlier, the ETS method presented in this paper is well suited for other combinatorial optimization problems. ETS-type algorithms form an emerging framework in computer programming that could challenge in the near future very sophisticated algorithms. New adaptations and refinements of the original idea are currently under study.

Acknowledgements

The author would like to thank Prof. Edward Silver for having introduced the problem to him and for his careful reading of an earlier version of this paper. He is also indebted to Phil Scheuer from the NHL for having provided the NHL data, to Pablo Moscato for his advices regarding the choice of a crossover operator and to Charles Fleurent for his helpful comments. Support of the *Fonds National de la Recherche Scientifique* (Grant No. FN 21-36173.92) is gratefully acknowledged.

References

[1] Booker L.,"Improving Search in Genetic Algorithms", Genetic Algorithms and Simulated Annealing, ed. Davis (Pitman, London and Morgan Kaufmann Publishers, Inc., 1987), 61-73.

[2] Costa D.,"On the use of some known methods for T-colorings of graphs", Annals of Operations Research 41 (1993), 343-358.

[3] Eglese R.W.,"Simulated Annealing: A tool for Operational Research", European J. of Operational Research 46 (1990), 271-281.

[4] Ferland J.A., Fleurent C.,"Computer aided scheduling for a sport league, INFOR 29-1 (1991), 14-25.

[5] Fleurent C., Ferland J.A., "Allocating games for the NHL using integer programming", Operations Research 41-4 (1993), 649-654.

[6] Fraser W., "The role of computer simulation in building the National Hockey League Schedule", IBM Canada Limited, Montreal, Quebec, Canada (1982).

[7] Goldberg D.E., "Genetic algorithms in search, optimization, and machine learning", Addison Wesley (1989).

[8] Glover F., "Future paths for integer programming and links to artificial intelligence", Computers and Operations Research 13 (1986), 533-549.

[9] Glover F., "Tabu search-part I", ORSA Journal on Computing 1 (1989), 190-206.

[10] Glover F., "Tabu search-part II", ORSA Journal on Computing 2 (1990), 4-32.

[11] Glover F., "Scatter Search and Star-Paths : Beyond the genetic metaphor", Working Paper, University of Colorado, Boulder (1993).

[12] Glover F.,Taillard E., de Werra D.,"A user's guide to tabu search", Annals of Operations Research 41 (1993), 3-38.

[13] Gonzales T., Sahni S.,"Open Shop Scheduling to minimize Finish Time", Journal of the Association for Computing Machinery 23-4 (1976), 665-679.

[14] Grefenstette J.,"Incorporating Problem Specific Knowledge into Genetic Algorithms", Genetic Algorithms and Simulated Annealing, ed. Davis (Pitman, London and Morgan Kaufmann Publishers, Inc., 1987), 42-60.

[15] Gromicho J., Paixão, Bronco I.,"Exact Solution of Multiple Traveling Salesman Problems",Combinatorial Optimization (New Frontiers in Theory and Practice), NATO ASI Serie F, Vol. 82 (1992).

[16] Hansen P., Jaumard B., "Algorithms for the maximum satisfiability problem", RUTCOR Research Report 43-87, Rutgers University (1987).

[17] Hertz A., "Finding a feasible course schedule using tabu search", Discrete Applied Mathematics 35 (1992), 255-270.

[18] Hertz A.,de Werra D., "Using tabu search techniques for graph coloring", Computing 39 (1987), 345-351.

[19] Holland J.H., "Adaptation in natural and artificial systems", Ann Arbor: University of Michigan Press (1975).

[20] Kirkpatrick S., Gelatt C.D., Vecchi M.P.,"Optimization by Simulated Annealing", Science 220 (1983), 671-680.

[21] Liepins G.E., Hilliard M.R.,"Genetic Algorithms: Foundations and Applications", Annals of Operations Research 21 (1989), 31-58.

[22] Moscato P.,"An introduction to population approaches for optimization and hierarchical objective functions: A discussion on the role of tabu search", Annals of Operations Research 41 (1993), 85-121.

[23] Mühlenbein H.,Georges-Schleuter M., Krämer O.,"Evolution Algorithms in Combinatorial Optimization", Parallel Computing 7 (1988), 65-88.

[24] Perusch M., "Simulated Annealing applied to a single machine scheduling problem with sequence dependent set up times and due dates", Research Report 86-84, Techische Universität Graz (1986).

[25] de Werra D.,Hertz A.,"Tabu search techniques: a tutorial and an application to neural networks", OR Spektrum 11 (1989), 131-141.