

Determining the Number of Games Needed to Guarantee an NHL Playoff Spot

Tyrel Russell and Peter van Beek

Cheriton School of Computer Science
University of Waterloo
{tcrussel,vanbeek}@uwaterloo.ca

Abstract. Many sports fans invest a great deal of time into watching and analyzing the performance of their favourite team. However, the tools at their disposal are primarily heuristic or based on folk wisdom. This paper provides a concrete mechanism for calculating the minimum number of points needed to guarantee a playoff spot in the National Hockey League (NHL). Along with determining how many games need to be won to guarantee a playoff spot comes the notion of “must win” games. Our method can identify those games where, if a team loses, they no longer control their own playoff destiny. As a side effect of this, we can also identify when teams get lucky and still make the playoffs even though another team could have eliminated them.

1 Introduction

Hockey fans are interested in knowing when their team clinches a playoff spot. This problem can be modelled as a satisfaction problem and solved using constraint programming [1]. However, if the team has not clinched a playoff spot, this method provides no information about how close a team is to earning a playoff position. The problem of determining how close a team is to clinching a playoff spot can be modelled as an optimization problem that determines the minimum number of points that is necessary to guarantee a spot. This bound on the number of points can also be used to determine when a team has no guarantee of making the playoffs and when a team has lost a crucial game and left destiny in the hands of another team. These factors are interesting to hockey fans and can be generalized to other sports with playoff structures, such as baseball and basketball.

We solve the problem using a constraint model and a mixture of techniques from constraint programming and operations research including network flows, constraint propagation and dominance constraints. We decompose the problem so that we can use a multi-stage approach that adds constraints at each stage if no feasible solution is found. The NHL determines positioning by points and, if tied by points, by several tie breaking conditions. The stages of the solver correspond to the extra constraints needed to determine the extra tie breaking conditions. The first stage uses a combination of enumeration and network flows to determine a tight lower bound on the points needed and whether there exists

a feasible solution using only points as a criterion. The second stage also uses network flows to check the first tie breaking condition. The third stage uses a backtracking constraint solver to determine if there are any solutions using the second tie breaking condition.

Our solver can determine the minimum number of points for a given team, at any point in the season, within ten minutes and, for dates near the end of the season, in seconds. In sports, analysts, reporters and coaches often refer to “must win” games. The method used in this paper can identify games where losing that game, the team puts its playoff aspirations into the hands of its opponents. While this does not mean the team will not qualify for the playoffs, it does mean that nothing the team does guarantees a playoff spot. We identify nine teams in the 2006-07 season that lost one of these “must win” games and found themselves in a position to earn a playoff spot again through the actions of their opponents. Several teams experienced this phenomena four times during the season.

To understand this paper, a brief description of the mechanics of playoff qualification in the NHL is introduced. Some related work is presented in Section 3. A formal definition of the optimization problem for determining the minimal number of points needed to guarantee a team makes the playoffs is presented. Afterwards, we introduce the concept of an elimination set and explain how this is used to determine tight lower bound values on the optimal value. Since the bound requires the relaxation of tie breaking conditions, we discuss how we can reincorporate those conditions using the same sets. Finally, the specific constraint model is introduced along with some optimizations in the form of exploited dominance and combined consistency checking.

2 The NHL Playoff System

Since the last expansion of the league in 2000, the NHL has consisted of thirty teams arranged into two conferences of fifteen teams. Each conference is broken into three divisions with five teams each. Each team in the NHL plays eighty two games with 41 home games and 41 away games. Eight teams make the playoffs from each conference and to earn a playoff position a team must either be a division leader or one of the five best teams in their conference not including division leaders, called wild card teams.

Positioning in the NHL is determined by one primary criterion and several secondary tie breaking criteria. The primary criteria is the number of points earned by a team. The two common secondary criteria are total wins and points earned against teams with the same number of points and wins. A third tie breaking criteria, which is the number of goals scored on opponents, is sometimes used to break ties between teams tied after the first three criteria are applied.

Like many North American sports, an NHL game must end in a win or loss. However, the NHL has a unique scoring system as there are points awarded for reaching overtime even if a team does not win the game. The game is separated into three twenty minute periods and, if teams are tied after sixty minutes, a

five minute overtime period. If teams are tied after overtime, there is a shootout to decide the winner. If the game ends during regulation time— the first sixty minutes— then the winner of the game is awarded two points and the loser earns no points. If, however, the game ends either during the overtime period, which is sudden death, i.e. ends when a goal is scored, or the shootout, which must conclude with a winner, then the winner still earns two points but the loser earns a single point in consolation.

3 Related Work

Russell and van Beek [1] created a constraint programming model for calculating whether a given team had clinched a playoff spot. However, the optimization of the decision model requires a relaxation of the dominance constraints. This relaxation of the constraints along with the increased search space leads to a significant increase in the execution time of the solver such that relatively late season instances could not be solved within a day.

Approaches for determining the minimum number of games needed to guarantee a playoff spot have been proposed for the Brazilian football championship [2] and in Major League Baseball [3]. These sports either have a simpler scoring model or a simpler playoff qualification method. Robinson [4] suggested a model for the NHL but his model did not allow for wild card teams or tie breaking conditions. As well, only a theoretical model was presented without experimental results. Gusfield and Martel [5] show that this problem is NP-Hard. They put forth a method for calculating bounds on when a team has been eliminated from the playoffs but their method only works for a single wild card team and a simple win-loss scoring model. Our technique uses a similar method but differs in approach as we must account for multiple wild card teams.

Wayne [6] introduced the concept of a constant that could be used to determine whether or not a team was eliminated from the playoffs. Specifically, he introduced the concept of lower bound constant W^* which denoted the minimum number of points needed to earn a playoff spot. Gusfield and Martel [5] show how this idea can be extended to include a single wild card team and multiple division leaders. In this paper, we will also discuss the existence of an upper bound constant which represents the minimum number of points needed to guarantee a playoff spot.

4 A Formal Problem Definition

To define the problem formally, certain concepts and notations must first be introduced. We denote the set of teams in the NHL as T . We denote the conference that team i belongs to as C_i and the division that team i belongs to as D_i . Table 1a shows the different time variables that we use to superscript the other feature-based variables. Table 1b shows the different features in the NHL and the notation for each feature based on the number and type of opponent. For each instance, there is a schedule and a date d_0 along with the results of games

Date	Notation	Feature	vs. j	vs. Opposite Conference	Total
Current	d_0	Points	$p_{i,j}^{d_t}$	$ocp_i^{d_t}$	$p_i^{d_t}$
End	d_e	Wins	$w_{ij}^{d_t}$	$ocw_i^{d_t}$	$w_i^{d_t}$
Generic	d_t	Overtime Losses	$ol_{ij}^{d_t}$	$ocol_i^{d_t}$	$ol_i^{d_t}$
		Games Remaining	$g_{ij}^{d_0}$	$ocg_i^{d_t}$	$g_i^{d_t}$

(a)

(b)

Table 1. Variable Notation **(a)** The variables representing the different dates under consideration. **(b)** The variables representing the current state of the results at a given time d_t .

prior to d_0 . We define a scenario S to be a completion of the schedule from d_0 by assigning wins, losses, and overtime losses to the games scheduled after d_0 . We refer to the maximum possible points that could be earned by a team i if they won all of their remaining games from a given date d_t as $mpp_i^{d_t}$. We refer to the maximum points over all teams $T' \subset T$ at a given time d_t as $\max^{d_t}(T')$ and the minimum points over all teams $T' \subset T$ at a given time d_t as $\min^{d_t}(T')$.

A team only qualifies for a playoff spot if they are a division leader or a wild card team. We define a division leader to be the team i that has the maximal points at the end of the season within their own division (i.e. $p_i^{d_e} = \max^{d_e}(D_i)$) and has better tie breakers than any team with equivalent points in their division at time d_e . We define a wild card team i to be any team that is not a division leader but has a $p_i^{d_e}$ greater than at least seven other teams in i 's conference that are also not division leaders.

Given team k , a given date of the season, d_0 , a given schedule of remaining games and given results up to d_0 in the season, a *Playoff Optimization Problem* is to determine the minimal number of points at the end of the season, $p_k^{d_e}$, such that there exists no scenario where k does not qualify for the playoffs as either the leader of the division or one of the five wild card teams. Note that we refer to the given team as either the elimination team or simply k for the remainder of the document.

5 Solution Overview

In this section, we provide an overview to the solver that we use to solve the optimization problem. In order to solve all instances, we use a multi-stage solver that applies different techniques at each stage. In the first stage, we enumerate all of the feasible elimination sets of teams (see Sec. 6) and derive a tight lower bound for the number of points needed. If $p_k^{d_0}$ is greater than the bound then they have already qualified and if $mpp_k^{d_0}$ is less than the bound then they can

no longer guarantee. If the bound falls in between those values then with each set that obtains our lower bound, we need to check each tie break condition to determine if this lower bound is a feasible number of points to guarantee a playoff spot. The second stage checks to see if the first tie break condition, wins, is enough to eliminate k . We do this by enumerating the possible win values and checking them with a feasible flow algorithm (see Sec. 7). If there is no feasible solution using only points and wins as criteria, the third stage again using a feasible flow algorithm to check if there are any sets where teams are tied in both points and wins (see Sec. 7). If there exists feasible tie breaking sets, we use a backtracking constraint solver to determine if one of the sets can eliminate k (see Sec. 8). If there are no solutions at this point then k can guarantee a playoff spot if they earn enough points to reach the bound.

6 Generating and Bounding the Elimination Sets

In this section, we define elimination sets and present a method for determining a bound on the points achievable by that set. To calculate the lower bound, we generate all sets of eight teams that could compose the three division leaders and five wild card teams. Each of these sets has the potential to eliminate k at some point bound. The largest bound over all of these sets forms a tight lower bound on the solution to our problem, differing by at most one point.

We define an *Elimination Set* as a set of eight teams from the same conference with at least one team from each division and does not include k . For each team i , they must either have $mpp_i^{d_0} > p_k^{d_0}$ or be the only team in the set from a division D_i such that $D_i \neq D_k$.

We define the bound of an elimination set, E , as the $\max(\min^{d_e}(E))$ under all scenarios S where either $p_k^{d_e} = \min^{d_e}(E)$ or $p_k^{d_e} = mpp_k^{d_0}$. The maximum bound over all elimination sets is a tight lower bound on the solution to the complete problem differing by at most one point.

6.1 Calculating the Bound

To calculate the bound for a given elimination set, we adapt an idea by Brown [7] using iterative max flows to solve a sharing problem. We implemented a similar algorithm that shares the games between the teams so that the worst team in the set has the most points possible. By constructing a flow graph that allows us to determine a feasible share, we iterate until a valid distribution of games is found. We start out with a possible bound and determine its feasibility. If the bound is not feasible, we update the bound and check the feasibility of the new bound.

In order to find the best bound, teams win as many points as possible. This means that every loss by a team in the elimination set is an overtime loss and teams in the elimination set win all of their games against teams that are not except k . We formalize the points earned by a team i under this situation as,

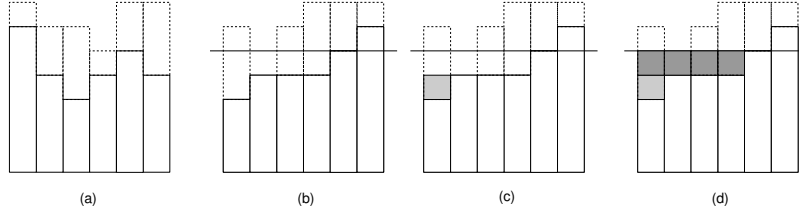


Fig. 1. The infeasible bound algorithm. **(a)** The original problem with 6 games remaining. The solid rectangles represents $p_i^{d_0}$ and the dashed rectangles represents $mpp_i^{d_0}$. The first step is to sort the teams by $p_i^{d_0}$. **(b)** shows the sorted teams with the $\min(mpp_i^{d_0})$ shown as the solid horizontal line. **(c)** We assign games to the teams with the least points. In this case, one game to the first team. **(d)** In the next iteration, four teams need games and we allocate four of the remaining five games. The bound is reached and the final solution has one game remaining.

$$p'_i = p_i^{d_0} + 2ocg_i^{d_0} + 2 \sum_{j \notin E \cup \{k\}} g_{ij}^{d_0} + \sum_{j \in E \cup \{k\}} g_{ij}^{d_0}. \quad (1)$$

These preprocessing steps are valid dominance relations as we are looking for the scenario where we get the maximum $\min^{d_e}(E)$ and these steps either increase the points of a team in E or leave them the same while not affecting the maximum possible points of the teams in E .

6.2 The Infeasible Bound

To determine the starting point for the lower bound, we solve a relaxation of the bound calculation where we relax the constraint that a specific number of games must be played between two teams. Instead, we consider all games as a pool of unplayed games with no assigned opponents and assign them to the worst team until the games are used or the $\min_{i \in E}(mpp_i^{d_0})$ is reached. Figure 1 shows an example bound calculation.

6.3 The Flow Network and the Bound

Once we have a starting point calculated by the infeasible bound algorithm, we are looking to find the first feasible bound when we include the constraints removed during the infeasible calculations. We formulate this as a feasible flow problem [8] with an artificial sink and source. These graphs look similar to the graphs constructed by Schwartz [9] in his paper on baseball elimination.

```

bound  $\leftarrow$  InfeasibleBound( $E, k$ );
repeat
  Needs  $\leftarrow$  CalculateNeeds( $E, k, \text{bound}$ );
  need  $\leftarrow$   $\sum_{i \in E} n_i$ ;
  G  $\leftarrow$  ConstructGraph(Needs);
  flow  $\leftarrow$  CalculateFlow(G);
  if flow < need then
    bound  $\leftarrow$  bound - 1;
until flow  $\geq$  need ;
return bound

```

Algorithm 1: This algorithm shows the steps for calculating the bound for a given elimination set, E . First, we generate an infeasible bound as a starting point. From that starting point, we generate needs to reach that bound for each team in the set. Then we check feasibility using a flow algorithm. If the flow meets the needs, we return the bound. Otherwise, we reduce the bound and iterate.

Every team in the elimination set and the team k has a need that must be incorporated into the graph. We define the need of a team i , n_i , to be $\text{bound} - p'_i$ (where bound is the current lower bound on points and p'_i is defined in (1)). The exception to this rule is k where the bound may be greater than mpp_k . In that case, we calculate n_k as $mpp_k - p'_k$. We use the p' values since we are still looking for best case results for the set $E \cup \{k\}$. A bound is feasible if the maximum flow in the graph is equal to the sum of the needs of the teams in the elimination set. If not, a new bound must be tried. Algorithm 1 describes the process by which the bound is calculated.

Once we know the needs for a given elimination set, it is relatively simple to construct the graph. An example graph is in Fig. 2 showing the variables and the associated capacities in the graph. We create two nodes s and t to be the source and sink, respectively. We add one node for each pair of teams in the set and one node for each team in the set. On top of this, we add an extra node that represents games from team k . Each node representing a pair of teams has three edges where one is an incoming edge from s with a capacity equal to the number of games between those two teams $g_{ij}^{d_0}$ and two are outgoing edges to the nodes for the teams with the same capacity as the incoming edge. There is also an edge from each node representing a team in the set to the sink node t with a capacity equal to the need of the node. Last, the node representing the games against k has an edge from the source with a capacity of $g_k^{d_0} - n_k$ and one link each to every team node with a capacity equal to the number of games played between them.

7 Win Values and Tie Breaking Sets

In this section, we describe how win values are used to determine if an instance has a solution and how to generate feasible tie break sets. Once we have a point bound and set of teams that could potentially reach that bound, we need to determine the possible values for the secondary criteria and only solve feasible

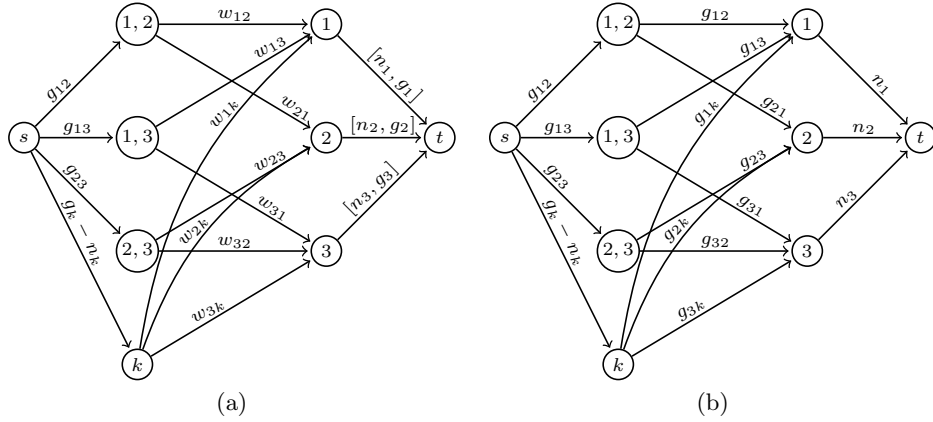


Fig. 2. (a) A variable representation of the values in the flow graph for a three team elimination set $(\{1, 2, 3\})$. The implementation of the graph uses the domain of the variables as the capacity bounds. (b) The capacities to determine feasibility for the flow graph. All variables are given their domain maximum. Since the flow is split in nodes $(1, 2)$, $(1, 3)$ and $(2, 3)$, a feasible flow is a valid assignment. If the max flow can saturate the needs of the teams then there is a feasible solution for this elimination set.

instances. We refer to the point bound calculated by the bounding algorithm as \mathbf{p} . This means that we need to determine if the elimination set can eliminate k with only wins or whether some teams need to be tied. We will use a modification of the original flow problem to determine both of these quantities. First, observe that if a team earns \mathbf{p} points then we have the following constraint,

$$(p_i^{d_e} = \mathbf{p}) \Rightarrow (\mathbf{p} - p_i^{d_0}) - g_i^{d_0} \leq w_i^{d_e} - w_i^{d_0} \leq \left\lfloor \frac{(\mathbf{p} - p_i^{d_0})}{2} \right\rfloor. \quad (2)$$

This constraint also holds true for k so we can determine a feasible range of wins for the elimination team given the elimination set, E , and the point bound \mathbf{p} . For each possible number of wins \mathbf{w} for k , we need to determine if the set can eliminate with that number of wins and, if not, which sets of teams can be tied.

Both of these tasks can be solved by checking for feasible flows on the same graph using slightly different needs in each case. We modify the graph for calculating point bounds by allowing k as a proper team on the right hand side and adding links directly to the nodes for games outside the set (see Fig. 3a). Games against opponents outside $E \cup \{k\}$ do not need to be won by any team in the set so they have no lower bound but those in the set must be won by one of the teams; therefore, they have a lower bound. We construct the graph so that each team that must be tied with k in wins has a lower and upper bound equal to their need. The need calculation for this graph is different than the point graph. Since both points and wins are both fixed, we get the following equation for calculating need,

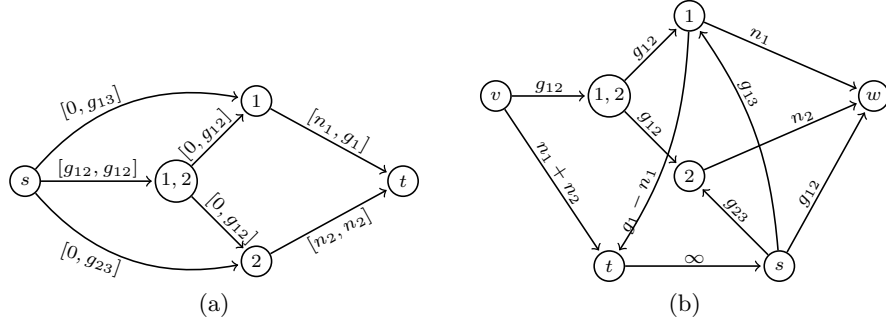


Fig. 3. (a) A network flow graph with three teams. Team 1 has a lower bound constraint on the number of wins and is in the elimination set and not in the tie break set, team 2 is in the tie break set and has a fixed number of possible wins, and team 3 is in neither the elimination set or the tie break set and has no bounds on either points or wins. (b) A flow graph transformed to remove the lower bound capacities. Two additional nodes are added v and w . A feasible flow exists in the original graph if the maximum flow is equal to the sum of the lower bounds on the original graph ($n_1 + n_2 + g_{12}$).

$$n_i = \begin{cases} \mathbf{w} - w_i^{d_0} & \text{if } (p_i^{d_e} = \mathbf{p}) \wedge (w_i^{d_e} = \mathbf{w}) \\ 0 & \text{if } (\mathbf{p} - p_i^{d_0}) - g_i^{d_0} < 0 \\ (\mathbf{p} - p_i^{d_0}) - g_i^{d_0} & \text{if } (\mathbf{p} - p_i^{d_0}) - g_i^{d_0} + w_i^{d_0} > \mathbf{w} \\ (\mathbf{p} - p_i^{d_0}) - g_i^{d_0} + 1 & \text{otherwise} \end{cases} \quad (3)$$

To check if the elimination set can eliminate k with wins, we set the needs to either 0, $(\mathbf{p} - p_i^{d_0}) - g_i^{d_0}$ or $(\mathbf{p} - p_i^{d_0}) - g_i^{d_0} + 1$ depending on the constraint and we set n_k to $\mathbf{w} - w_i^{d_0}$. For teams tied with k in wins, we introduce a tie break set. We define a *Tie Break Set* as any subset of the teams in C_k where every team can reach both the point bound \mathbf{p} and the win bound \mathbf{w} exactly. We test all subsets by setting the need of teams in the tie break equal to $\mathbf{w} - w_i^{d_0}$.

Since our graph has minimum and maximum capacities on the edges, we transform the graph into a different max flow problem as described by Ahuja et al. [8]. The transformation can be seen in Fig. 3. Once we have checked the wins tie breaker with the flow graph and determined which sets of nodes can be tied in both points and wins, we determine if any of those tie break sets can eliminate k with points against teams that are tied. We model this problem as a satisfaction problem and solve it using backtracking search as described in the next section.

8 The Decision Problem

In this section, we describe the constraint model used to determine if the final tie breaks eliminate k . Once we have fixed the elimination set (E), point bound (\mathbf{p}),

win bound (\mathbf{w}) and tie break set (TB), we verify this combination eliminates the team k . We are examining possible scenarios of the remaining games in the schedule. Specifically, we are interested in the wins ($w_{i,j}$) and overtime losses ($ol_{i,j}$) as these are the two factors that affect the points and hence the outcome of a given scenario. These variables have domains ranging from zero to the number of games being played between the two teams i and j , $g_{i,j}$. We break the teams into four mutually exclusive classes to help describe our model.

$$\begin{aligned} A &= \{i \mid i \in E \wedge i \notin TB\} & C &= \{i \mid i \notin E \wedge i \in TB\} \\ B &= \{i \mid i \in E \wedge i \in TB\} & D &= \{i \mid i \notin E \wedge i \notin TB\} \end{aligned}$$

8.1 The Model

There are four major constraints to this model. Each of which is modified slightly depending on which class a given team belongs. Constraint (4) represents the constraint that each of the teams must either meet or exceed the bounds depending on their class. These rules are derived from the NHL tie breaking rules. Constraint (5) represents the constraint that each game must have a winner. The exception to both of these constraints are those teams in D . These teams are not restricted by the bounds and thus we can ignore any game where they are playing other teams in D . We also need to constrain the number of overtime losses so that the team does not earn more overtime losses than losses. This constraint is reflected in (6). Lastly, we must deal with constraints on games played against teams in the opposite conference. Teams in A can win these games freely, teams in D can lose them freely and teams in B and C can win them depending on the constraints applied in (4). We define these constraints explicitly in (7).

$$\begin{aligned} p_i^{d_e} &> \mathbf{p} \vee (p_i^{d_e} = \mathbf{p} \wedge w_i^{d_e} > \mathbf{w}) && \text{if } i \in A . \\ p_i^{d_e} &= \mathbf{p} \wedge w_i^{d_e} = \mathbf{w} \wedge \\ \left(2 \sum_{j \in TB} w_{ij}^{d_e} + \sum_{j \in TB} ol_{ij}^{d_e} \right) &> \left(2 \sum_{j \in TB} w_{kj}^{d_e} + \sum_{j \in TB} ol_{kj}^{d_e} \right) && \text{if } i \in B . \\ p_i^{d_e} &= \mathbf{p} \wedge w_i^{d_e} = \mathbf{w} && \text{if } i \in C . \quad (4) \\ \forall_j \quad w_{ij}^{d_e} + w_{ji}^{d_e} &= g_{ij} && \text{if } i \notin D . \\ (\forall_{j \in A} \quad w_{ij}^{d_e} = w_{ij}^{d_0} \wedge w_{ji}^{d_e} = w_{ji}^{d_0} + g_{ij}^{d_0}) \wedge \\ (\forall_{j \in B \cup C} \quad w_{ij}^{d_e} + w_{ji}^{d_e} = w_{ji}^{d_0} + g_{ij}^{d_0}) \wedge \\ (\forall_{j \in D} \quad w_{ij}^{d_e} = w_{ij}^{d_0} \wedge w_{ji}^{d_e} = w_{ji}^{d_0}) &&& \text{if } i \in D . \quad (5) \\ \forall_j \quad w_{ij}^{d_e} + ol_{ij}^{d_e} &= w_{ij}^{d_0} + ol_{ij}^{d_0} + g_{ij}^{d_0} && \text{if } i \in A . \\ \forall_j \quad w_{ij}^{d_e} + ol_{ij}^{d_e} &\leq w_{ij}^{d_0} + ol_{ij}^{d_0} + g_{ij}^{d_0} && \text{if } i \in B \cup C . \\ \forall_j \quad ol_{ij}^{d_e} &= ol_{ij}^{d_0} && \text{if } i \in D . \quad (6) \end{aligned}$$

$$\begin{aligned}
ocw_i^{d_e} &= ocw_i^{d_0} + ocg_i^{d_0} \wedge ocol_i^{d_e} = ocol_i^{d_0} && \text{if } i \in A \text{ .} \\
ocw_i^{d_e} + ocol_i^{d_e} &\leq ocw_i^{d_0} + ocol_i^{d_0} + ocg_i^{d_0} && \text{if } i \in B \cup C \text{ .} \\
ocw_i^{d_e} &= ocw_i^{d_0} \wedge ocol_i^{d_e} = ocol_i^{d_0} && \text{if } i \in A \text{ .}
\end{aligned} \tag{7}$$

8.2 Updating Dominance During Search

As the search progresses, it is often possible to force the assignment of certain games. The most important dominance is to notice that only win variables within the tie break and elimination set need to be set via search. Once those variables have been set, all that remains is to ensure teams meet or exceed \mathbf{p} and \mathbf{w} and to make sure teams trying to beat k earn as many of their necessary overtime losses within the tie break set and k wins as many of them as possible out of the tie break set. These dominances lead to a correct solution and makes sure teams in B earn as many points within the tie break as possible.

Another opportunity is when teams have satisfied (4). Specifically, once a team in A has met the conditions of (4), they may give points to other teams in A without any consequences. Another dominance is that once a team in the tie break set has achieved both \mathbf{p} and \mathbf{w} they must lose any remaining games in regulation time.

8.3 Pruning Values from Constrained Teams via Flow Manipulation

As mentioned in Sec. 7, the feasibility of the tie break set depends on whether there exists a max flow equal to the needs of the teams in the flow graph represented by Fig. 3. An important observation that can be made is that any feasible flow is a valid assignment of the win variables of the teams in the elimination and tie break sets. We can prune the variables within the solver by attempting to update an already existing flow to contain a specific test value using a method adapted from Maher et al.[10]. If there is a flow that contains the value then there is a support for that value and that value is kept. If not, then we prune the value from the domain of the variable. The idea is similar to the idea used in the Ford-Fulkerson algorithm. However, in our case, we are trying to find a path not from s to t but from j to i . We must repeat the update at most d times where d is the size of the domain of w_{ij} and w_{ji} .

To reduce the practical complexity of the algorithm, we reduce the residual graph to only those components that will be updated. In a graph containing a feasible flow, the edges out of v and into w are completely saturated. Since any modification must also be a feasible flow, these edges must remain saturated and any modification should not alter these edges. The other reduction that we can make to the graph depends on the symmetry between nodes representing teams and the links to their matched games. This allows us to remove the nodes representing the matched games and link the nodes directly together.

Example 1. Examine Fig. 4b and note that in the residual graph links into v and out of w are saturated and can be removed. Also observe that the edge

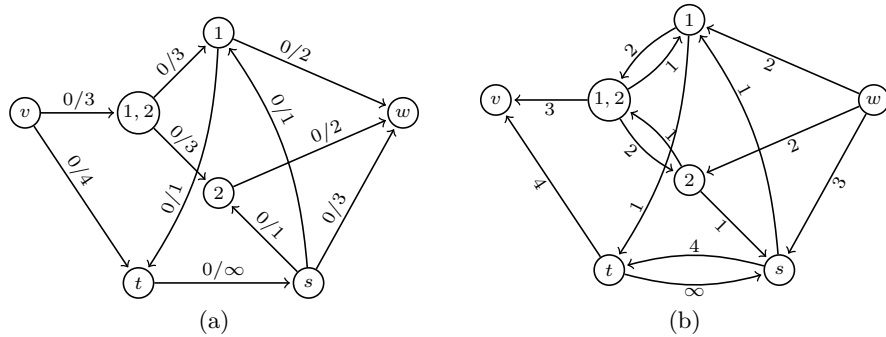


Fig. 4. (a) An example flow graph for three teams where Team 1 must earn between 2 and 3 games, Team 2 must earn exactly 2 games and Team 3 is unbounded. (b) The residual graph containing a maximum flow.

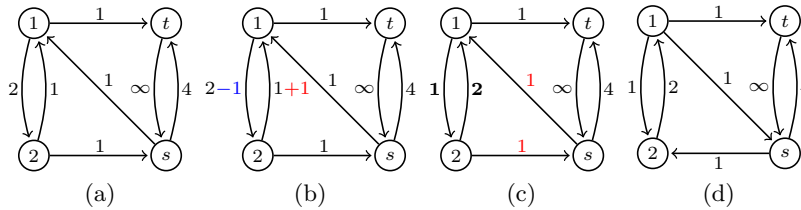


Fig. 5. Reduced Pruning Graph. (a) shows the reduced residual graph of Figure 4. In (b), we reduce the link between nodes 1 and 2 and increase the link between nodes 2 and 1, which ensures the constraint that the flow between them equals some mutual capacity. (c) shows the path that is found from node 2 to node 1 correcting the imbalance of residual flow into node 2. Once a path is found, the flow is redirected and the opposite edges are updated by the change. In this case, edges that once had zero residual capacity are increased and are not shown. (d) shows the new stable solution showing support for the assignments of $w_{12} = 1$ and $w_{21} = 2$.

from node 1 to node (1, 2) is the same as the edge from node (1, 2) to node 2. Therefore, we can remove node (1, 2) and directly link (2, 1). Figure 5 shows the reduced pruning graph along with a single variable update.

9 Results

We implemented the solver in C++ using the Boost Graph Library [11] for the feasible flow calculations and ILOG Solver[12] to solve the final constraint model. In order to test our solver, we used the 2006-07 season results to calculate the minimum points needed to clinch a playoff spot. Table 2 shows the results of those calculations. In total, determining the bound for all 30 teams on all 181 game days of the 2006-07 NHL season (5430 problems) took a little over 46 hours.

Solver Stage & Result	Number of Instances (/5430)
Solved via Enumeration	1212
Solved via Win Checks (Positively)	2249
Solved via Win Checks (Negatively)	1524
Solved via Backtracking Search (Positively)	338
Solved via Backtracking Search (Negatively)	107

Table 2. The counts of problems solved via the various stages of the solver. Positively solved instances means a solution was found and the bound must be increased. Negatively solved instances means that bound was valid for that instance. Any problem without a definitive solution was passed to the next phase of the solver.

Each individual instance took less than ten minutes to calculate the bound and those problems near the end of the season, where the results matter the most, were calculated in seconds. We note that our enumeration techniques solves 1212 of the 5430 of the problems and when we add first level tie breaking with wins we solve a further 3773 problems, which makes up about 92% of the problems. However, the remaining 8% problems require a backtracking constraint solver to calculate the final number. Also, note that in 47% of the total instances, which amounts to 61% of the instances not solved directly by enumeration, the answer differs from the initial lower bound given by enumeration.

We plot the result against both the current points of the team and maximum possible points of the team. If the result is greater than the maximum possible points, then the team is no longer able to guarantee a playoff spot. Note that we only calculate the bound for the maximum possible points plus one as any greater number is also infeasible as the team does not have sufficient points to reach the bound. If the result is equal to the number of points needed by the team then that team has clinched a playoff spot. Figure 6 shows the result calculated for Toronto and Pittsburgh. This graph has several interesting features. First, the number of points earned never reaches the minimum points needed to clinch a playoff spot. Hence, Toronto did not make the playoffs. Also note that Toronto placed themselves in a position where they could not guarantee a playoff spot and got lucky four times. In other words, they lost a must win game five times during the 2006-07 season while Pittsburgh was never in that situation. Another interesting feature is that we can see, in both graphs, the bound on points, 145, needed at the start of the season to guarantee a playoff spot.

Table 3 shows an overview of the results of the 2006-07 NHL season in terms of the minimum points needed to guarantee a playoff spot. One interesting observation that can be made from this table is that of the nine teams that got a second chance only two of those teams ended up earning a playoff spot. As well, of those seven teams, two of them had four chances to make the playoffs after losing a must win game. Another interesting note is that St. Louis could not guarantee a playoff spot after only the sixty-fourth game day and never recovered during the final one hundred and eighteen game days.

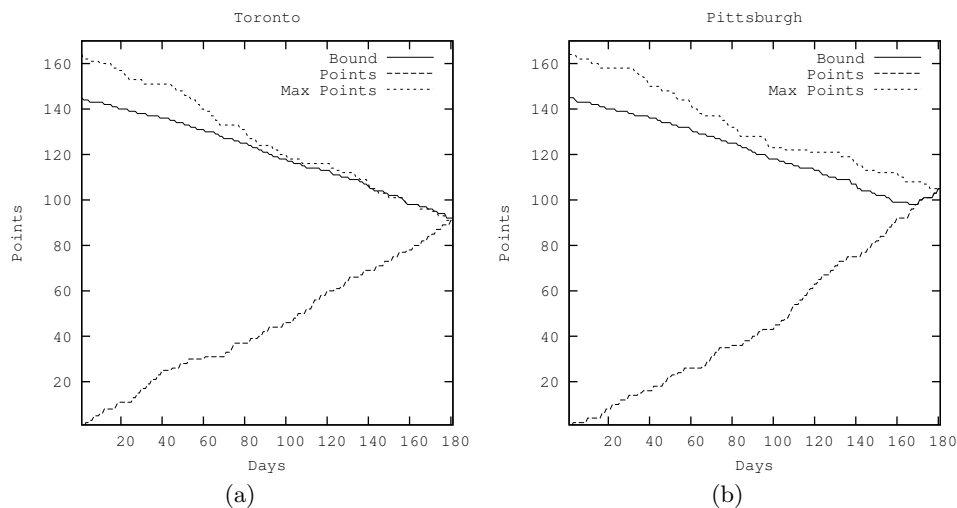


Fig. 6. The minimum number of points needed by Toronto and Pittsburgh to guarantee a playoff spot in the 2006-07 NHL season.

10 Conclusion

As the season winds down, the fans of the NHL are interested in knowing how far their team is from clinching a playoff spot. We present a method for calculating the minimum number of points that must be earned in order to ensure that the team reaches a playoff spot. We perform this calculation efficiently by using a multi-stage solver that combines enumeration, flow network calculations and backtracking search. As a corollary to this calculation, we can also determine when the team is in danger of losing control of their destiny. These games, often described by coaches as “must win” games, can be identified by their loss reducing the maximum possible points to below the bound of the team. We identified nine different teams in the 2006-07 NHL season that lost control of their fate and then gained that control back through mistakes by their opponents. We also noted that only two of these teams took full advantage of this situation and clinched a playoff spot. Of the nine teams which experience this event, three of them experienced it four times.

Our solver used a decomposition of the problem to allow us to effectively apply several different strategies in several stages to ensure a quick solution to the problem. The first stage of the problem enumerated all of the possible elimination sets and used a feasible flow calculation to determine the maximum points earned by each team. The second stage again applied a feasible flow calculation on a modified graph that incorporated both points and wins. And finally, on the most difficult problems, we used a backtracking constraint solver to analyze the final tie break conditions. Using this decomposition strategy, we found that we could calculate each instance in a matter of minutes.

Feature	Value	Team(s)
Earliest Day where a Team could not Guarantee	64 days	St. Louis
Most Days where a Team could not Guarantee	118 days	St. Louis
Most Times a Team got Lucky	4	Toronto, Boston and NY Rangers
Number of Teams that got Lucky and Earned a Spot	2	NY Islanders and NY Rangers
Number of Teams that got Lucky but Failed to Earn a Spot	7	Toronto, Boston, Washington, Carolina, Edmonton, Phoenix and Columbus

Table 3. Shows some of the features that can be highlighted by calculating the minimum number of points needed to guarantee a playoff spot.

The results of this work could be applied to other sports. One area that seems to be missed entirely is basketball, especially NBA basketball, where that league shares many similarities with the NHL. The tie breaking conditions vary slightly and the NBA uses a simpler scoring model with only wins and losses.

References

1. Russell, T., van Beek, P.: Mathematically clinching a playoff spot in the NHL and the effect of scoring systems. In: Proceedings of the 21st Canadian Conference on Artificial Intelligence. (2008)
2. Ribeiro, C.C., Urrutia, S.: An application of integer programming to playoff elimination in football championships. *International Transactions in Operational Research* **12** (2005) 375–386
3. Adler, I., Erera, A.L., Hochbaum, D.S., Olinick, E.V.: Baseball, optimization and the world wide web. *Interfaces* **32** (2002) 12–22
4. Robinson, L.W.: Baseball playoff eliminations: an application of linear programming. *Operations Research Letters* **10** (1991) 67–74
5. Gusfield, D., Martel, C.E.: The structure and complexity of sports elimination numbers. *Algorithmica* **32** (2002) 73–86
6. Wayne, K.D.: A new property and a faster algorithm for baseball elimination. *SIAM Journal on Discrete Mathematics* **14** (2001) 223–229
7. Brown, J.R.: The sharing problem. *Operations Research* **27** (1979) 324–340
8. Ahuja, R.K., Magnanti, T.L., Orlin, J.B.: *Network Flows: Theory, Algorithms and Applications*. Prentice Hall (1993)
9. Schwartz, B.: Possible winners in partially completed tournaments. *SIAM Review* **8** (1966) 302–308
10. Maher, M., Narodytska, N., Quimper, C.G., Walsh, T.: Flow-based propagators for the sequence and related global constraints. In: Proceedings of the 14th International Conference on Principles and Practice of Constraint Programming. (2008)
11. Siek, J., Lee, L.Q., Lumsdaine, A.: *Boost Graph Library: User Guide and Reference Manual*. Addison-Wesley (2001)
12. ILOG S.A.: *ILOG Solver 4.2 user’s manual* (1998)