Lessons Learned from Modelling the NHL Playoff Qualification Problem

Tyrel Russell and Peter van Beek

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

Abstract. The modelling of complex problems tends to be most effective when modelling is calibrated using a concrete solver and modifications to the model are made as a result. In some cases, the final model is significantly different from the simple model that best fits the constraints of the problem. While there are several projects that are attempting to create black box solvers with generic modelling languages, for the moment the modelling processes is intimately tied to the solving and search procedure. This paper looks at the changes to the simple model and the search procedures that were made to create an efficient solution to the NHL Playoff Qualification problem. The approaches for improving the model could be extended to other applications as the techniques are not specific to this problem.

1 Introduction

The specific choices for modelling a given problem have a significant impact on the ability to solve that problem efficiently [22]. Compounding the difficulties in this task is the observation that, unlike modern SAT solvers, CP solvers are not yet black box solvers and solving a problem with a CP solver can require significant interplay between the solving and modelling phases. In this paper, we discuss the difficulties associated with modelling a real world combinatorial optimization problem, the NHL Playoff Qualification Problem [18, 19]. Unlike our previous work where we present solutions to such problems, in this paper we discuss the issues arising from not only modelling the problem but the specific solving methods which drove the changes to the original model.

This paper highlights some of the challenges arising from modelling a combinatorial optimization problem as a constraint programming problem and illustrates some techniques that have been used to solve the problem. Specifically, we discuss how enumeration of logical constraints into decomposed models can allow for more direct propagation of constraints. Disjunctive constraints, which underlie implication constraints, have efficient GAC propagators but the resulting pruning is still quite weak in practice [2, 7, 11, 24]. We identify the conditions needed to apply this technique to solve other problems. We also re-illustrate the effectiveness of solving relaxations of the original problem, which can give good bounds on the original problem [6]. In our case, we used enumeration to generate

models where a simple relaxation of the model allows for a polynomial solution to the relaxed models. The addition of symmetry and dominance constraints to the model also had a significant impact on the speed at which solutions were obtained.

One other technique that we used in solving these problems was the decomposition of mutually exclusive disjunctive constraints into solving phases. The observation that further decomposition allowed for simple polynomial solutions in some cases led to a significant speed up in the time needed to solve the problem. A combination of further enumeration and constraint programming was used to solve these decomposed phased models.

2 The Abstract NHL Qualification Problem

The NHL Qualification Problem is a winner determination problem where the goal is to determine the number of points needed to just earn a playoff spot rather than win the league. The NHL is a two part competition broken into a regular season and a playoff tournament. The winner of the playoffs wins the Stanley Cup, which is regarded as the most prestigious cup in North American hockey. Since a team must qualify for the playoffs during the regular season, it shows why determining whether you have earned a spot in the playoffs is so important.

The NHL has thirty teams that are broken up into two conferences of fifteen teams. Each conference is further broken into three divisions each with five teams. A playoff spot is earned if either the team is the top ranked team in their division or they earn one of the five wild card spots for their conference. A wild card spot is a playoff spot given to a team that did not win their division but was one of the top five non-division winners in their conference. The ranking is decided on several criteria applied in order: most points, most wins, most points earned against teams that have equal points and wins and most goals scored by the team against other teams. In this work we do not consider the last tie breaking criteria as the number of goals scored is unbounded in any given game.

Definition 1. *Given a remaining schedule of games left to play, the results up to a given point of the season, i.e. points and wins earned by teams so far, and a distinguished team k, an NHL Playoff Qualification Problem is the problem of determining the number of points needed by k such that if they earn that number of points there exists no scenario, i.e. a completion of the remaining games, such that they do not qualify for the playoffs.*

This problem is known to be NP-Hard when there are m spots that make the playoffs from each conference or when there are wild card spots [13, 5]. The current playoff structure used by the NHL is an instance of both classes of problems. Similar problems have been solved for Brazilian soccer [17], which has $m = 8$ spots that make the playoffs, and MLB baseball [1], which has a single wild card per conference.

Before we introduce the model, we first present some notational definitions. We denote the current day or date of the season as d_0, the end date of the season as d_e, and an arbitrary date as d_t, $d_0 \leq d_t \leq d_e$. For every team, we have a win variable, w, and overtime loss variable, ol, for each opponent and subscript each variable with the indices ij which denotes that team i achieved the result over team j. We also superscript the variable with the given date, especially to denote the difference between wins earned up to the current date, $w_{ij}^{d_0}$, and wins earned at the end of the season under some scenario, $w_{ij}^{d_e}$. The points earned by the team i against an opponent j is the weighted sum of the wins, worth two points, and the overtime losses, worth a single point, denoted $p_{ij}^{d_t} = 2w_{ij}^{d_t} + ol_{ij}^{d_t}$. We also denote the number of games remaining for a team i against a team j at date d_t as $g_{ij}^{d_t}$ and the sum of games remaining for team i against all teams as $g_i^{d_t} = \sum_j g_{ij}^{d_t}$. The total points earned at a given date d_t is simply the points earned against all opponents, denoted $p_i^{d_t} = \sum_j (p_{ij}^{d_t})$. Sets C_i, D_i, and TB_i are used to denote the set consisting of all teams in the same conference as team i, the set consisting of all teams in the same division as team i and the set of teams tied with team i in both points and wins.

A basic model includes the constraints that the points of k must be maximized (1), the total number of wins must be equal to the number of games (2), the number of wins and overtime losses must be less than the number of games (3), there must be eight teams, three division leaders and five wild cards, which either have more points or have better tie breakers ((4) and (5)).

$$\max p_k^{d_e} \ , \tag{1}$$

$$\forall_{ij} \quad w_{ij}^{d_e} + w_{ji}^{d_e} = w_{ij}^{d_0} + w_{ji}^{d_0} + g_{ij}^{d_0} \ , \tag{2}$$

$$\forall_{ij} \quad w_{ij}^{d_e} + ol_{ij}^{d_e} \leq w_{ij}^{d_0} + ol_{ij}^{d_0} + g_{ij}^{d_0} \ , \tag{3}$$

$$\begin{aligned}
\forall_i \quad & \left(p_i^{d_e} > p_k^{d_e}\right) \\
\vee \ & \left(p_i^{d_e} = p_k^{d_e} \wedge w_i^{d_e} > w_k^{d_e}\right) \\
\vee \ & \left(p_i^{d_e} = p_k^{d_e} \wedge w_i^{d_e} = w_k^{d_e} \wedge \sum_{j \in TB_k^{d_e}} p_{ij}^{d_e} > \sum_{j \in TB_k^{d_e}} p_{kj}^{d_e}\right) \\
\vee \ & \Big[\forall_{d \in D_i} \left(p_i^{d_e} > p_d^{d_e}\right) \\
& \vee \left(p_i^{d_e} = p_d^{d_e} \wedge w_i^{d_e} > w_d^{d_e}\right) \\
& \vee \left(p_i^{d_e} = p_d^{d_e} \wedge w_i^{d_e} = w_d^{d_e} \wedge \sum_{j \in TB_i^{d_e} \wedge j \in D_i} p_{ij}^{d_e} > \sum_{j \in TB_i^{d_e} \wedge j \in D_i} p_{dj}^{d_e}\right)\Big] \\
\Leftrightarrow \ & b_i = 1 \ ,
\end{aligned} \tag{4}$$

$$\sum_{i \in C_k} b_i \geq 8 \ . \tag{5}$$

We introduced an indicator variable b_i which is 1 if and only if i has more points, better tie breakers or is a division leader. If there exists eight or more teams where this is true then team k needs $p_k^{d_e} + 1$ points to guarantee a playoff spot. The problem with this model is that the core of the model is a combination of implication and disjunction constraints. However, like other real world problems, the NHL is of bounded size and, therefore, it is sometimes practical to enumerate some of the variables or constraints of the model. In the next sections, we will discuss how we used enumeration and decomposition to find feasible solution to models with (4).

3 Enumerating the Set of Implication Constraints

Implication constraints are widely used in constraint models. Examples of applications that include implication constraints include protein folding [3], configuration problems [8], telecommunication feature subscription [10], pipeline planning and scheduling [14], and the travelling salesman problem with time windows [16]. However, while they easily capture the constraints inherent in many applications, they provide relatively weak propagation [11].

For example, take a simple constraint, "if the power supply has a power rating of 150 watts then only use low power components," that might be found in a configuration problem. This constraint, which we will simplify as $p = 150 \Rightarrow use\_low\_power$, has very weak propagation. The constraint cannot be propagated until the left hand side has been instantiated to a specific value or the right constraint has been falsified. This can be seen easily if the constraint is rewritten as a disjunction, $\neg(p = 150) \vee (use\_low\_power)$. No propagation occurs until either a 150 watt power supply is selected or a high power component is selected. This problem is still apparent even when generalized arc consistency is enforced on the disjunction [11]. Note that in the example as long as another power supply is possible or the use_low_power constraint can be satisfied then no propagation occurs unless the two constraints shared a variable.

Enumeration is a common operations research and constraint programming technique for decomposing a problem into more manageable sub-problems. The goal is to enumerate the values of specific variables to decompose the constraint graph or propagate a singleton value. This differs from the goal here which is to allow constraints to be posted earlier to increase the propagation available to the solver. Examples of applications which use this decomposition technique include sports scheduling problems [15], instruction scheduling [12], and diagnostics [20].

Tree decompositions can be used in combination with enumeration. When converting from a regular cyclic hypergraph to an acyclic graph, the solutions of the sub-problems generated can be enumerated to generated a completely acyclic constraint graph [4]. Rymon introduced a technique for enumerating the power set of possible sets systematically in a best first fashion [20]. This is similar to the technique used here but again the enumeration used by Rymon was on variable values where as, in this work, it is on constraint implications which does

not necessarily result in any singleton variable propagation but rather allows the implied constraint to be posted earlier than in the simple model.

Given the limited size of an NHL conference, it is possible to enumerate all of the different ways that the indicator variables b_i could be set in the model. If we only look at one side of the logical equivalence, we end up with a constraint that looks like,

$$b_i \Rightarrow \left(p_i^{d_e} > p_k^{d_e}\right) \vee (p_i^{d_e} = p_k^{d_e} \wedge \dots \quad . \tag{6}$$

In the literature, we find there are a number of different applications that have similar structures with a single indicator variable implying a constraint on the model [10, 14, 16]. If we take a generic implication, $p \Rightarrow q$, we can rewrite this constraint as a simple disjunction $\neg p \vee q$. If p is a simple indicator variable which needs a specific value, an arc consistent propagator will correctly prune no values from the variables in the constraint q unless p has be set to the specific value under consideration. While this behaviour is logically correct, in practice it means that the constraint q may not propagate any values until late in the search tree when it may have been the case that p must have been the specified value in every solution. Note, however, that if p is not set to the specific value then the constraint q can either be true or false. Constraint (4) is a constraint of logical equivalence and does not necessarily have this property. However as long as (5) is satisfied, we can substitute (6) for (4) and search for solutions to the b_i variables as it does not strictly matter if there is nine or more teams better than k as long as there is eight. In other words, if another team j should be better than k by the constraint we do not need to enforce that b_j is true as long as the sum of the b_i variables is greater than or equal to eight.

With this in mind, we can look at solutions of the b_i variables which lead to immediate propagation of the right hand side of Constraint (6). We do not have to strictly look at sets of nine or more because if a larger set has a feasible solution then a smaller subset would have a feasible solution as long as no constraint on the indicator variables (i.e. (5)) is violated. We can generalize this result in some cases as long as we have a set of implication constraints with indicator variables of the form, $p_i \Rightarrow q_i$, and feasibility of the constraints on the indicator variables can be established. In the following result, we look at specific configurations of the indicator variables. We define an assignment of the indicator variables p_i as the set of indicator variables P_T which are set to true and all other indicator variables set to false. For each indicator variable $p_i \in P_T$, there is a corresponding q_i constraint which must be satisfied in the model and we denote the set of constraints that must be satisfied under an assignment P_T as Q_T. We define the set of constraints that can be affected by fixing the value of an indicator variable p_i, directly or by chaining through other variables in the constraints, as the set $I(p_i)$ and the set of all constraints affected by setting a set of indicator variables P_T as $I(P_T) = \bigcup_{p_i \in P_T} I(p_i)$. We say that a constraint c can be affected by chaining if there exists a sequence of constraints from c to a constraint containing the indicator variable p_i such that each constraint in the sequence shares at least one variable with its predecessor and successor in the sequence.

Lemma 1. *Given a set of implication constraints, $p_i \Rightarrow q_i$, if there exists a solution where a set of p_i variables, P_T, set to true yields a feasible solution then any subset $P_T^- \subseteq P_T$ of variables set to true yields a feasible solution as long as the constraints in $I(P^T)$ have a feasible solution for the assignment of variables in P_T^-.*

Proof. Assume that there is a solution where the indicator variables in the set P_T are set to true and the other variables set to false. Now assume that the model is infeasible for some subset of P_T, P_T^-. This means that some constraint in the model that was not violated under the assignment P_T has now been violated under the assignment P_T^-. However, the constraints in the model for P_T^- are a subset of the constraints in the model under the assignment P_T and the model is strictly more relaxed, the only constraints which could have caused the model to become infeasible are those in the set $I(P_T)$. □

There exists another dominance in the set of indicator variables. If the constraints enforced by an assignment is infeasible then any set containing that assignment must also be infeasible. If the the set of constraints Q_T is infeasible under the assignment P_T, then any set P_T^+ such that $P_T \subseteq P_T^+$ is also infeasible.,

Lemma 2. *If the set of constraints Q_T associated with an assignment P_T are infeasible, then any set P_T^+, such that $P_T \subseteq P_T^+$, will enforce an infeasible set of constraints.*

Proof. Since the set of constraints Q_T^+ enforced by P_T^+ contains the an infeasible subset of constraints Q_T, the entire set of constraints is infeasible. □

Sets of indicator variables which have been instantiated form a distinctive lattice structure where each set of size k has a number of parents of $k+1$ indicator variables (see Figure 1). Using Lemma 1, we know that if there is a solution for some value $k \geq 8$ in the lattice then we immediately have a solution for a set of size 8. This can be useful later when testing tie breaking criteria which can be easier on smaller sets of implication constraints. Lemma 2 tells us that if we find any solution of any size that is infeasible then one need not look at any set which contains that subset. This allows us to prune sets of indicator variables which cannot be mutually satisfied.

Since there are fewer solutions at the top and the bottom of the lattice, it can be beneficial to test these sets as the most benefit is gained from pruning. However, the small sets are the ones least likely to be infeasible and the large sets are the least likely to be feasible. Therefore, it can be useful to search various locations in the tree to achieve maximum pruning and lead to a solution as early as possible. However, in the next section, we discuss how almost all of the subsets are dominated using bounding and thus do not need to be examined.

4 Bounding and Pruning the Sets

It is well known that tight lower bounds are useful in solving combinatorial optimization problems. The most common use is in branch and bound search where

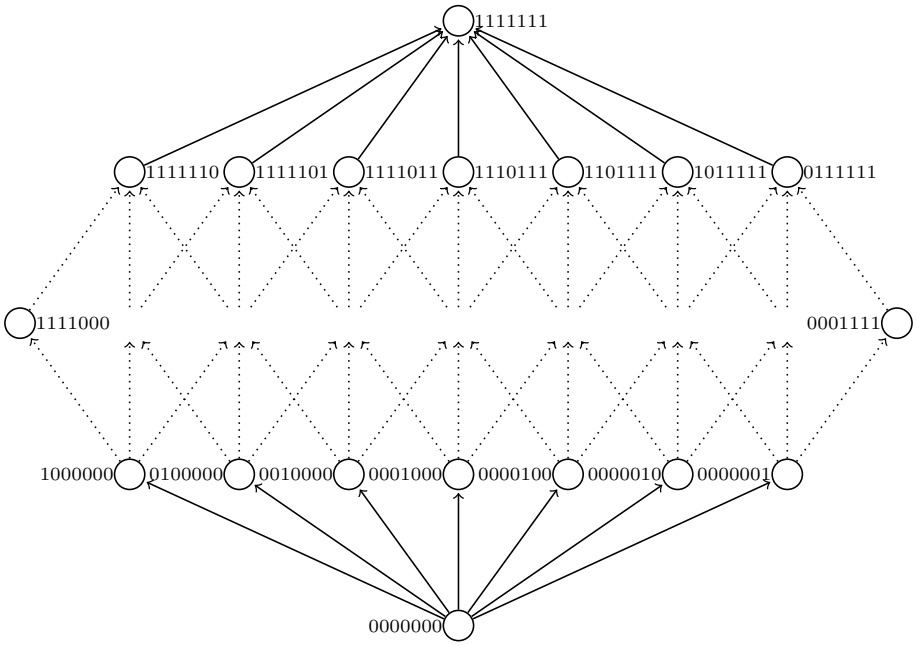

Fig. 1. Shows the lattice of possible sets of indicator variables that can be generated for the implication constraints. Within this graph, it is possible to find chains of dominance and infeasibility.

lower and upper bounds are used to prune infeasible regions of the search space. Mixed integer programming relies heavily on this technique due to the existence of an easily obtainable polynomial relaxation, the LP relaxation. In a constraint programming context, an integer optimization function may be represented as a constrained integer whose domains may be pruned by good bounds.

Since the NHL Playoff Qualification problem is an optimization problem, not all of the feasible solutions lead to an optimal solution. Lemma 1 tells us that if there is a feasible solution with more than eight constraints enforced, there is a feasible solution with exactly eight implication constraints enforced. In this section, we show that we can bound the quality of the assignment of indicator variables and that a smaller feasible set always has a value at least as large as a super set. We also show that using bounds pruning we can remove many of the sets with feasible solutions for eight teams. We integrate this idea into our dominance rules from the indicator variable lattice to help prune sets of indicator solutions.

We construct a bound by taking a set of indicator variables and we relax Constraint (6) to,

$$b_i \Rightarrow \left(p_i^{d_e} \geq p_k^{d_e}\right) \quad . \tag{7}$$

Constraint (7) forms the relaxed form of the original constraint where we are considering only the first tie breaking condition. In other words, we only consider the world where a set is at least as good k. This is a lower bound on the original problem since the distinguished team may need one more point to actually break ties which are relaxed using (7). The following lemma formalizes this notion.

Lemma 3. *The basic constraint model given an assignment P_T and where Constraint (6) is replaced by Constraint (7) provides a tight lower bound which is at most one less than the actual solution for the given set of indicator variables if a solution exists.*

Proof. The solution obtained by substituting the relaxed constraint finds the maximum value of $p_k^{d_e}$ such that each team $i \in P_T$ has at least as many points. Note that if k earns one more point then there must exist one team in P_T that cannot simultaneously obtain $p_k^{d_e} + 1$ points along with k or the solution was not a true maximum. Therefore, if k earns one more point than the relaxed bound, they necessarily qualify and could qualify at the lower bound if they are better on tie breaks. □

Using Lemma 3, we know that if we can calculate the points for a specific set of indicator variables we have both a tight lower and a tight upper bound on the value for that given set of indicator variables. When checking for feasibility, we noted that a feasible solution for a large set of variables meant there was a feasible solution for all subsets as long as the assignment of indicator variables did not violate the constraints. Now, we show that the smaller sets necessarily have a bound value that is at least as large as a set containing the small set.

Lemma 4. *Given two sets of fixed indicator variables, P_T^+ and P_T, such that $P_T \subseteq P_T^+$ the solution for the NHL Qualification model under the assignment P_T has a solution whose value is at least as large as the value of the solution to the NHL Qualification model under the assignment P_T^+ as long as there is a feasible solution to both models.*

Proof. Assume there is a feasible solution to both models and that the solution under the model enforcing the constraints in Q_T has a value that is less than the value of the solution to the model enforcing the constraints in Q_T^+. However, the constraints in model under the assignment P_T^+ include all the constraints, possibly with some extra constraints, that are in the model with the under the assignment P_T. Therefore, any solution to the P_T^+ model is a solution to the P_T model as long as (5) is satisfied under P_T which is given since we assume there exists a feasible solution to the P_T model. Therefore, we have a contradiction and the solution to model under the assignment P_T must have a value at least as large as the value of the solution to the model under the assignment P_T^+. □

Lemma 4 states that smaller sets are optimal if they are feasible and Lemma 2 shows that any super set of a small set is infeasible if the small set is infeasible. As a corollary, this means that it is sufficient to only look at the smallest sets

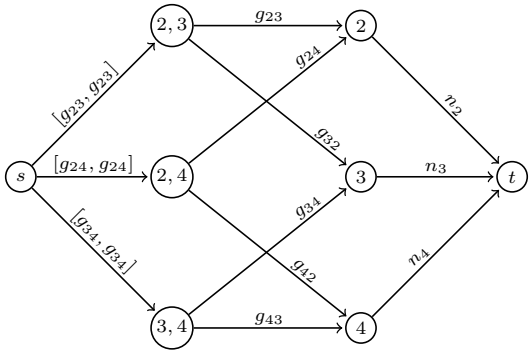

Fig. 2. This figure shows the flow network for a four team problem where the first team earns $w_1^{d_e}$ wins. This means that the other three teams must earn less than $w_1^{d_e}$. Assuming that no team has not already earned that many wins, the number of wins that the team cannot exceed is $n_i = w_1^{d_e} - w_i^{d_0}$. Assume that the lower bound capacity is zero unless a range is given.

that satisfy both the constraints on the indicator variables and the constraints on the rest of the model.

We have discussed why finding bounds on the values is an effective technique for pruning different sets of models but we have not discussed how this is done specifically for this problem. Schwartz [21] showed in 1966 that a similar problem, the winner determination problem, could be solved in polynomial time for a win-loss scoring model. Kern and Paulusma [9] showed that the same approach could be used for other scoring models if they are normalized. McCormick [13] and Gusfield and Martel [5] showed that playoff determination problems are often NP-Complete. However, if we enumerate the solutions, we can use a method similar to that used by Kern and Paulusma [9]. One thing that differs is that we cannot make as strong of assumptions about the play of the distinguished team. As well, Kern and Paulusma's model does not explicitly state how to deal with the factors removed during normalization when they are reincorporated into the model. The basic method described by Schwartz [21] uses a flow network to partition remaining games to individual teams. If there exists a feasible flow, which represents an assignment of wins, then there exists a scenario where the distinguished team earns the most points and they have not be eliminated. Figure 2 shows an example of a flow network constructed using Schwartz's method.

The NHL uses an unusual $\{(0,2),(1,2),(2,1),(2,0)\}$ model for scoring games which normalizes to the win-loss model. However, the normalization procedure assumes two things that are not necessarily true in the problem we are describing. First, it assumes that $(1,2)$ and $(2,1)$ always dominates $(0,2)$ and $(2,0)$, respectively. The second issue that is not described is how to deal with the extra points removed when the model is reduced from $\{(1,2),(2,1)\}$ to $\{(0,1),(1,0)\}$.

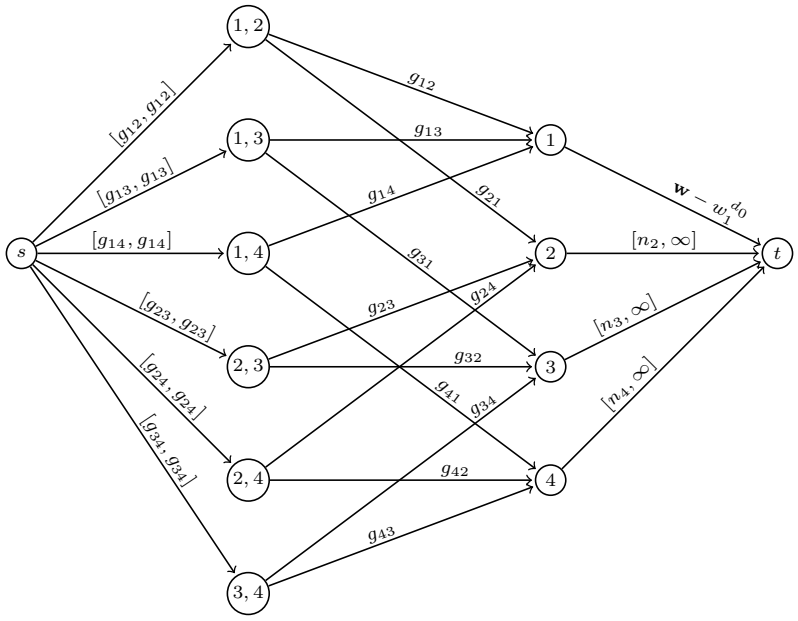

Fig. 3. This figure shows the same flow network as Fig. 2 but here we must include team 1 since we cannot make assumptions about the remaining games for team 1. Assume this graph is calculated for $\mathbf{w}$ wins. Now we also want the teams to reach or exceed the wins of team 1. The number of wins is calculated in the same manner, $n_i = \mathbf{w} - w_i^{d_0}$, but forms a lower bound in the graph.

These points for each team except the distinguished team can be directly incorporated into flow graph.

In a playoff qualification problem, we are looking for scenarios on the edge of feasibility. Those solutions where the distinguished team k may earn a playoff spot but earning one more point will surely earn them a playoff spot. These situations do not occur in the problems introduced by Schwartz as they did not consider tie breaking. Therefore, we are looking for a scenario where a set of teams denoted earn at least as many points as k. As well, given the enumeration, we know which teams we want to reach the bound so our graph need only include second layer nodes if the team is in the set or is k as all others have no bound associated with them. An example of the new graph associated with the playoff qualification problem is shown in Fig. 3.

5 Decomposition of Mutually Exclusive Disjunctions: Phased Solver Approach

Recall that (4) is a combination of a logical implication and a disjunction where even maintaining arc consistency may prune very few domains until late in the

search. However, mutually exclusive disjunctions provide a simple method for decomposing the model as only one of the conditions can be true at any given time. One method for dealing with mutually exclusive disjunction is through branching on exclusive choices in the disjunction as described by Van Hentenryck [23, pg. 169]. Our approach, of enumerating and using different solvers, differs because we can easily check feasibility of combinations of disjunctions and determine when a problem is hard and can be reserved until later, if needed. Observe in (4) that the first three conditions have a mutually exclusive structure,

$$\begin{aligned} &\left(p_i^{d_e} > p_k^{d_e}\right) \vee \left(p_i^{d_e} = p_k^{d_e} \wedge w_i^{d_e} > w_k^{d_e}\right) \\ \vee &\left(p_i^{d_e} = p_k^{d_e} \wedge w_i^{d_e} > w_k^{d_e} \wedge \sum_{j \in TB_k^{d_e}} p_{ij}^{d_e} > \sum_{j \in TB_k^{d_e}} p_{kj}^{d_e}\right) . \end{aligned} \tag{8}$$

Each constraint naturally excludes the others being true and thus provides a simple decomposition point as each term in the constraint being true means the other terms cannot be true. These terms form a tie breaking constraint which means that the relaxed version of the first, where we allow equality, would include all solutions to the second and third and the relaxed version of the second with equality would include the solutions to the third.

These observations allow for a phased strategy where we take the enumeration of constraints and extend it to not just the relaxed constraint but all three terms listed in (8). In the first phase, we determine the bounds on the enumerated sets as described in Section 4 and keep only those teams with the highest lower bound, $\mathbf{p}$. Using only the models with eight implication constraints set, we post the first disjunction instead of the entire constraint. If there is a solution where every team earns more than the bound then we return the solution and we are done. Note it is possible for this to happen as the bound may be limited by the maximum number of points earned by k. Then, we relax the constraint by adding equality and check for feasibility. If there is now a solution then there may be a solution under tie breaking and we store that set to check later. If there is no solution to the relaxed problem, then we know there is no solution for that set and we discard.

In the second phase, we have only sets where teams can reach the point bound $\mathbf{p}$ and we know from the results of the first phase that there exists no scenario where every team earns more points than the bound. Therefore, we determine if in those scenarios where teams just reach the bound, the tied teams earn more wins. Observe that k has a certain number of earned wins $p_k^{d_0}$ initially and some maximum and minimum possible given the number of points needed to reach the bound. The difference between the maximum and minimum is due to the possibility of overtime losses which earns the team an extra point. This range of wins can be used to prune the number of different sets that we examine. For example, if k can earn between 21 and 24 wins to reach a point bound of 53 points and each team in the set needs to reach at least 53 points then many of those teams would not reach or exceed the point total without also earning

more or less wins than is possible for k given the possible unevenness in the overtime losses. Therefore, if we enumerate the possible win totals for k for each set then we can determine for each win value which teams could be tied. By examining all sets of tied teams and enforcing the second disjunction only for those teams while leaving the remainder constrained by just the first disjunction, we can solve the problem efficiently. Note we discard any scenario where some team cannot reach or exceed either the point bound or the win bound, when tied. Again as in the first phase, if there is a solution without equality then stop with a solution. If there is a solution with equality, then save it for the the next level of tie breaking and if no solution then there is no feasible solution for a given $\mathbf{p}$.

The third phase is similar to the second phase except that we are looking for sets of teams that have equal number of points and wins and all other sets can be ignored. Here, however, we must consider the entire set of teams and not just sets of size eight as the dominance that worked for the first two rounds no longer holds. It is possible for there to be a solution where more than eight teams are tied with k but not with eight due to the fact that the teams left out may help everyone but k and thus create a circumstance where a larger set is necessary. Increased propagation is possible in this phase by adding constraining tied teams to the known point and win bounds.

The first two phases can be solved with network flows once the enumeration has occurred but the third stage is still hard and requires that a constraint programming solver be used for the last stage.

6 Symmetry and Dominance

Another common modelling method is to add additional constraints, either statically or dynamically, to remove symmetries and dominances from the solution. One reason to add these constraints is to avoid the search of identical solutions and thus reduce the tree search. However, it is well known that this technique does not always payoff as the static variable ordering technique sometimes conflicts with the branching heuristic and the overhead from the dynamic checks and constraint posting may be too large. We use this technique combined with the initial enumeration to remove symmetries and dominances that could not be easily detected without the enumeration.

Once we know the point bound and have fixed the set of implication constraints that will be active in the model, there are symmetric and dominated solutions that can be removed by static symmetry breaking constraints. In this section, we describe some of the symmetry techniques that can be used by enumerating the features that make nearly symmetric solutions symmetric.

The most obvious is that we can discard those fixed sets of implication constraints which do not have an upper bound equal to the point bound. Those solutions are dominated by those sets with a better lower bound.

There is very little true symmetry in the NHL Playoff Qualification problem as each team often has a different number of games remaining, points, wins and

	Enumeration	Bounding	Phased Solver	Symmetry Breaking
Timeouts (300 s)	5075 [1]	4217	2	0
Latest Date of Failure	Apr 4	Apr 4	Oct 5	N/A
Earliest Problem Solved	Mar 18	Dec 5	Oct 4	Oct 4

Table 1. Results from the 2006-07 season. Instances are solved for every game day, working in reverse order from the end of the season (Apr 8) to the beginning of the season (Oct 4). There are a total of 5430 instances.

different opponents for their remaining games. However, by using the phased approach combined with enumeration we are able to identify some classes of teams where symmetries can be applied to reduce the search space of the problem.

Specifically, the enumeration tells us when constraints will be used and thus when disjunctions do not have to be posted. Therefore, we know when we can force certain assignments as they always lead to a solution if there exists a solution. The first is that teams that have no constraints, i.e. the indicator variable is not true, need not win any games. The second is that in many cases the specific overtime losses can be fixed especially for teams that have no constraints.

7 Experimental Results

The combination of the various techniques generates a set of models for each problem that can be easily solved. Each component builds on the techniques used previously. The initial enumeration is used to deal with the implication constraints and the cardinality constraint on the indicator variables. Once the sets have been enumerated, it is easy to prune the sets since bounds can be calculated efficiently. The phased solver method takes the idea of enumeration and extended that to disjunctions as well as implication constraints. Lastly, once the enumeration has fixed both the points and wins earned by the distinguished team, it is possible to fix many of the other variables due to symmetry.

Table 7 shows the improvements of the various techniques. The solvers were implemented using C++ using ILOG Solver 4.2 and the Boost Graph Library. Implementing a sum of variables indexed by a set of indices from a set variable proved infeasible using ILOG Solver 4.2 so the simple model was not implemented in its pure form. From the results, we see that implementing enumeration allows some problems to be solved but it must be combined with other techniques to solve the problem. The largest improvement gain was from using enumeration and decomposition to create a phased approach. Symmetry breaking constraints were added to solve the last remaining problems.

[1] Due to time constraints, timeouts for Enumeration are estimated. Once a team timed out ten times, it was extrapolated that all of the remaining instances would time out. This is a relatively safe assumption since the instances get larger towards the beginning of the season and the gap between the bound and the current points increases.

8 Discussion

The lessons learned from modelling the NHL Qualification Problem can be applied to other models. The direct posting of implication constraints can be helpful but note that in the worst case there is an exponential number of sets to consider. This method works best when the constraints on the indicator variables can be easily checked, such as in our case where there is a single cardinality constraint, and the total number of implication constraints is relatively small. It is also important for the posted constraint to prune the domains; i.e., if the right hand side of the implication provides no propagation then the enumeration is purely overhead without any benefit.

Enumeration can also be useful, as in this case, because it reveals problem structure. By carefully enumerating specific parts of the model, the model can be decomposed which made it more amenable to other efficient techniques such as flow networks. Another example where this technique has been used before is in tree decompositions [4].

Static symmetry breaking constraints can be a very useful tool but only when coupled with a heuristic that does not conflict with the choices. We took care to ensure that our symmetry breaking constraints did not conflict with the heuristic we were using.

9 Conclusion

This paper is a case study in modelling a real world optimization problem. In the process of modelling the problem, enumeration and decomposition was used to modify the simple model which encoded the constraints into a set of models that could be easily solved by themselves. The enumeration of the implication constraints created models where constraints could be propagated earlier. The enumeration also allowed several more techniques to be applied, most notably, bounding and the decomposition of mutually disjunctive constraints. The final decomposition uncovered symmetry that was not present in the initial model and the static symmetry breaking constraints helped improve the efficiency.

References

1. I. Adler, A. L. Erera, D. S. Hochbaum, and E. V. Olinick. Baseball, optimization and the world wide web. *Interfaces*, 32:12–22, 2002.
2. F. Bacchus and T. Walsh. Propagating logical combinations of constraints. In *Proc. of the 19th Intl. J. Conf. on AI*, 2005.
3. R. Backofen. Using constraint programming for lattice protein folding. In *Proc. of the 3rd Pacific Symposium on Biocomputing*, 1998.
4. R. Dechter and J. Pearl. Tree clustering for constraint networks. *Artificial Intelligence*, 38:353–366, 1989.
5. D. Gusfield and C. E. Martel. The structure and complexity of sports elimination numbers. *Algorithmica*, 32:73–86, 2002.

6. J. Hooker. Logic, optimization, and constraint programming. *Informs Journal of Computing*, 14:295–321, 2002.
7. C. Jefferson and K. Petrie. Efficient propagation of disjunctive constraints using watched literals. In *Proc. of the 7th Intl. Workshop on Constraint Modelling and Reformulation*, 2008.
8. U. Junker. Preference programming for configuration. In *Proc. of the 4th Workshop on Configuration*, 2001.
9. W. Kern and D. Paulusma. The computational complexity of the elimination problem in generalized sports competitions. *Discrete Optimization*, 1:205–214, 2004.
10. D. Lesaint, D. Metha, B. O'Sullivan, L. Quesada, and N. Wilson. Solving a telecommunications feature subscription configuration problem. In *Proc. of the 14th Intl. Conf. on the Princ. and Prac. of CP*, 2008.
11. O. Lhomme. Arc-consistency filtering algorithms for logical combinations of constraints. In *Proc. of the 10th Intl. Conf. on the Princ. and Prac. of CP*, 2004.
12. A. Malik, M. Chase, T. Russell, and P. van Beek. An application of constraint programming to superblock instruction scheduling. In *Proc. of the 14th Intl. Conf. on the Princ. and Prac. of CP*, 2008.
13. S. T. McCormick. Fast algorithms for parametric scheduling come from extensions to parametric maximum flow. *Operations Research*, 47:744–756, 1999.
14. A.V. Moura, C.C. de Souza, A. A. Cire, and T. M. T. Lopes. Planning and scheduling the operation of a very large oil pipeline network. In *Proc. of the 14th Intl. Conf. on the Princ. and Prac. of CP*, 2008.
15. G. Nemhauser and M. Trick. Scheduling a major college basketball conference. *Operations Research*, 26:1–8, 1998.
16. G. Pesant, M. Gendreau, J.-Y. Potvin, and J.-M. Rousseau. An exact constraint logic programming algorithm for the traveling salesman problem with time windows. *Transportation Science*, 32:12–29, 1998.
17. C. C. Ribeiro and S. Urrutia. An application of integer programming to playoff elimination in football championships. *Intl. Trans. in Operational Research*, 12:375–386, 2005.
18. T. Russell and P. van Beek. Mathematically Clinching a Playoff Spot in the NHL and the Effect of Scoring Systems. In *Proc. of the 21st Conf. of the Canadian Society for Computational Studies of Intelligence*, 2008.
19. T. Russell and P. van Beek. Determining the number games needed to guarantee an NHL playoff spot. In *Proc. of the 6th Intl. Conf. on the Integration of AI and OR Techniques in CP for Combinatorial Optimization Problems*, 2009.
20. R. Rymon. Search through systematic set enumeration. In *Proc. of the Third Intl. Conf. on the Princ. of Knowledge Representation and Reasoning*, 1992.
21. B. Schwartz. Possible winners in partially completed tournaments. *SIAM Review*, 8:302–308, 1966.
22. H. Simonis. Models for global constraint applications. *Constraints*, 12:63–92, 2007.
23. P. Van Hentenryck. Constraint Satisfaction in Logic Programming. *MIT Press*, 1989.
24. J. Wurtz and T. Muller. Constructive disjunction revisited. In *In Proc. of the 20th German Conf. on AI*, 1996.