Optimization in Constraint Reasoning about Repeating Events

Robert A. Morris
Lina Khatib
Florida Institute of Technology
Melbourne, FL 32901
email:{morris,lina}@cs.fit.edu

Abstract

The effective manipulation of temporal information about periodic events is required for solving complex problems such as long range scheduling or querying temporal information. Furthermore, many problems involving repeating events require the optimization of temporal aspects of these events, e.g., minimizing makespan in job-shop scheduling. This paper contains a framework for representing and solving reasoning problems in which temporal aspects of repeating events are to be optimized. This framework rests upon three foundations: recent work on the characterization of repeating events [Morris et al.*, 1996], the temporal CSP framework for processing constraints, [Dechter* et al.*, 1991], and the semiring generalization of CSPs [Bistarelli* et al.*, 1995]. The semiring allows values to be associated with consistent solutions generated from the specification; the values are computed from functions that provide interpretations of the optimizing constraints.*

1 Introduction and motivation

A long-term scheduling problem might involve scheduling more than one occurrence of the same event. For example, in telescope observation scheduling [Bresina, 1994], telescope time for the purpose of observing time-varying phenomena (e.g. eclipsing binary stars) is requested by an astronomer. An astronomer's scientific agenda (e.g., to fill out a light curve for a binary star system), imposes various constraints; for example, on the number of observations and on the number of nights between successive observations. Thus, an astronomer might request that a given number of repeated observations (specified by an ideal and minimum occurrence count) be executed within a given time window with a given time gap between observations. The ideal gap (in days) is specified either with a fixed gap length or a gap probability distribution (in order to reduce aliasing in the data or to determine the period of a recently discovered variable star). An example of a gap probability distribution would be expressed as "gaps should be randomly selected with a uniform probability from the set { 0 days, 1 day, 2 days }".

This example illustrates that a repeating event can be characterized in terms of a set of properties, among them the number and duration of each occurrence of the event, the duration of the gap between successive occurrences, and the length of time during which all occurrences must complete. The objective of this paper is to present a framework for representing and reasoning about periodic events within the CSP framework. This framework rests upon three foundations: first, the work of [Morris *et al.*, 1996] on representing repeating events within a CSP setting; second, the *semiring CSP* (SCSP) generalization of the CSP framework [Bistarelli *et al.*, 1995]; and third, the Temporal CSP framework [Dechter *et al.*, 1991] for effective reasoning about temporal relationships among events. The resulting formulation allows for problems like those found in the telescope observation domain, in which repeating events have constraints that are to be optimized, to be solved.

The remainder of this paper describes a representation of repeating events as a set of intervals with a temporal structure (section 2); a language for specifying constraints on repeating events (section 3); an extension of this language for formulating and solving optimization problems within the semiring CSP framework (section 4); a brief discussion of further extensions for specifying and reasoning about binary relations between repeating events (section 5); and a brief survey of previous work on computational aspects of repeating events (section 6).

2 Temporal profiles of repeating events

A single repeating event can be described as a collection of intervals with a *temporal structure*. The temporal structure assumed in the remainder of

this paper is adhered to by any finite set of non-overlapping intervals. By the *instantiation* of a repeating event, we mean the set of times during which the *sub-intervals* of a repeating event start or end. For example, where I is a repeating event, one instantiation of I would be expressed by the set $I = \{I_1 = [3, 15], I_2 = [20, 30], I_3 = [75, 84]\}$.

In scheduling problems, it is common to specify constraints on the set of possible instantiations for a set of events. Often these constraints can be described in terms of restrictions on the durations or other temporal properties of these events. This can be applied as well to constrain the instantiations of the sub-intervals of a single repeating event.

Matrices can be used effectively to represent temporal information about repeating events. Each matrix will be referred to as a *profile* of the repeating event. 5 profiles can be distinguished: 4 in terms of end-point combinations (end-start, start-start, start-end, and end-end) and 1 is qualitative. The qualitative profile of any finite set of m non-overlapping sub-intervals is an $m \times m$ matrix with three "regions". For example, the qualitative profile of the repeating event I described above is the 3 by 3 matrix

$$qual[I] = \begin{vmatrix} = & b & b \\ bi & = & b \\ bi & bi & = \end{vmatrix}$$

where $qual[I][i, j]$ is the Allen relation between I_i and I_j, the ith and jth sub-intervals of I. The three regions of a qualitative profile of non-overlapping intervals are: a lower-left triangle of bi relations, an upper-right triangle of b relations, and a diagonal of $=$.

A *gap profile* measures temporal distances between end-points and start-points. There are two gap profiles that are represented as matrices $g_{s,e}$ and $g_{e,s}$. For example,

$$g_{s,e}[I] = \begin{vmatrix} 12 & 27 & 81 \\ -5 & 10 & 64 \\ -60 & -45 & 9 \end{vmatrix}$$

is a gap profile of I. Where $s(I_i), e(I_i)$ are, respectively, the start and end times of sub-interval I_i, $g_{s,e}[I][i, j] = e(I_j) - s(I_i)$. Thus, negative values indicate that I_i starts after I_j ends. The value of the profile $g_{e,s}$ is defined in a similar way where $g_{e,s}[I][i, j] = s(I_j) - e(I_i)$. Thus,

$$g_{e,s}[I] = \begin{vmatrix} -12 & -27 & -81 \\ 5 & -10 & -64 \\ 60 & 45 & -9 \end{vmatrix}.$$

Similarly, a *start period profile* $p_{s,s}$ measures the distances $s(I_j) - s(I_i)$ between start-points of pairs of sub-intervals, whereas an *end period profile* $p_{e,e}$ measures the distances between pairs of end-points.

In a quantitative profile of finite non-overlapping events, each row is a sequence of either increasing or decreasing values, as is each column. Furthermore, when the row values increase (read left-to-right), column values decrease (read top-down), and vise versa. Any matrix which satisfies these properties will be said to be *admissible*.

A constraint on a repeating event can be viewed as specifying a subset of the set of admissible profiles. It is typical in reasoning problems about repeating events to specify constraints over only part of a complete profile, rather than the complete set of end-point relationships. For example, it is common to specify gap constraints between *successive* sub-intervals, rather than between arbitrary pairs. Similarly, it is common to specify period constraints on the specific distance $s(I_{i+1}) - s(I_i)$, rather than between arbitrary pairs of starting points. The remaining discussion will focus on the representation of such constraints, which will be referred to as *diagonal constraints*, for they tend to restrict values along or near the diagonal of a profile.

3 CSP formulation

A CSP is based on a set of variables $X = X_1, X_2, \ldots X_n$ each with a domain $D_i, i = 1 \ldots n$, and a set of constraints $\mathcal{C}$, each of which is a relation (set of tuples) defined on a subset of X. A *solution* to a CSP is a complete assignment to elements $X_i \in X$ of values from D_i such that each constraint in $\mathcal{C}$ is satisfied. In a *temporal* CSP, the variables stand for events, the domains for possible time units (typically, intervals or points) and the constraints describe allowed temporal orderings or durations.

Associated with each repeating event I will be a set of variables representing profile or other temporal information. Variables for any subset of the 5 profiles can be used. In the remaining examples, the following two variables will be used for profile information:

- d_I: *sub-interval duration*, i.e., $e(I_i) - s(I_i)$;
- g_I: *gap duration between successors* i.e., $s(I_{i+1}) - e(I_i)$;

In addition, two variables are introduced to allow constraints to be specified on the *number of sub-intervals* in I (n_I) and the *extent* of I (e_I), by which is meant the duration from the start of the earliest sub-interval to the end of the latest.

Definition 1 A specification of a repeating event I is a pair $(V, \mathcal{C})$, where $V = \{n_I, d_I, g_I, e_I\}$ is a set of variables, each with an integer domain. Corresponding to the variables is a set $\mathcal{C}$ of *domain constraints* of the following form:

- sub-interval number: $l_n \leq n_I \leq u_n$;
- sub-interval duration: $l_d \leq d_I \leq u_d$
- gap duration: $l_g \leq g_I \leq u_g$
- extent: $l_e \leq e_I \leq u_e$.

Constraints for sub-interval or gap duration are implicitly constraints on the values of a set of intervals; hence their first-order formulation is a universal quantification. For example the logical form of

$$l_d \leq d_I \leq u_d \text{ is}$$

$$\forall I_i \in I \; e(I_i) - s(I_i) \in [l_d, u_d].$$

Example 1 A specification for a repeating event I:

- sub-interval number: $2 \leq n_I \leq 6$;
- sub-interval duration: $2 \leq d_I \leq 6$
- gap duration: $3 \leq g_I \leq 10$
- extent: $25 \leq e_I \leq 50$.

This example clearly specifies that I is to consist of between 2 and 6 sub-intervals, each with a duration of between 2 and 6 time units, with gaps between successive sub-intervals in the range of between 3 and 10 time units, and I is to take no longer than between 25 and 50 time units.

Given a specification, a solution to this reasoning problem assigns values to each variable in V which satisfies each domain constraint and, in addition, preserves the temporal structure imposed on the repeating event, i.e., the result is a finite sequence of non-overlapping intervals. We call this the problem of *profiling the diagonal of a repeating event.*

A general method for profiling the diagonal of a repeating event is to transform this problem into a binary Temporal CSP (TCSP), in particular, that of a *Simple Temporal Problem* (STP) [Dechter *et al.*, 1991]. This can be accomplished as follows. First, by a *concretization* of a repeating event I with specification $(V,\mathcal{C})$ is a network whose nodes consist of end-points of sub-intervals of I, and whose arcs are labeled with intervals obtained from $\mathcal{C}$. A concretization is simply a binary TCSP.

Figure 1 visualizes the relationship between the specification of a repeating event and a concretization, using the previous example. The rectangle stands for a repeating event, and the looping arc is labeled with the conjunction of domain constraints in the specification. We say that an assignment to n_I *triggers* a concretization, in the sense that it defines the size of the resulting network. This triggering role is depicted in the picture by the labeled arc between the rectangular node, and the concretization. Because of the assignment $n_I = 2$, there are 4 nodes in the concretization, one for each end-point of the sub-intervals of I. The arcs between nodes are labeled by the corresponding constraint intervals.

The problem of profiling a repeating event can thus be solved by finding a consistent concretization of the repeating event, then solving the resulting STP to find a consistent *atomic labeling* of the edges of the network. As described in [Dechter *et al.*, 1991]), since the constraints specified involve a single interval, such STPs can be solved simply *via* a mapping into a structure called a *distance graph*;

$$n_I \in [2,6] \wedge d_I \in [2,6] \wedge g_I \in [3,10] \wedge e_I \in [25,50]$$

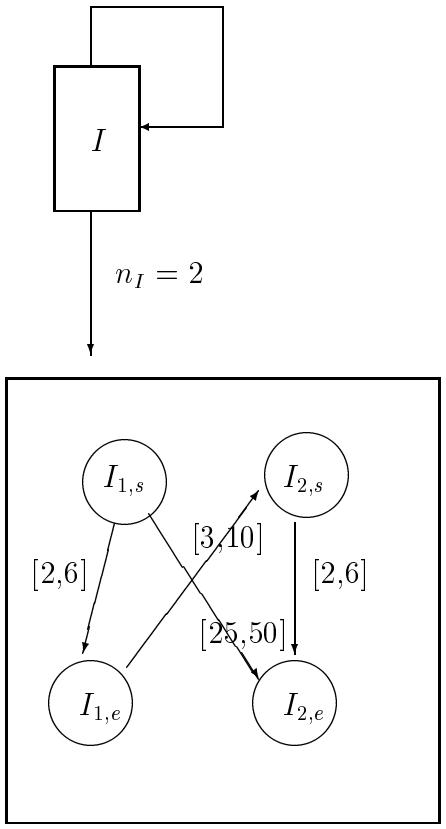

Figure 1: Specification and concretization of a repeating event

a consistent atomic labeling from such a graph can be assembled in $O(n^3)$ time.

To apply optimizing conditions to constraints, a mechanism needs to be in place to evaluate such atomic labelings. Toward this end, we first present a uniform representation of a consistent labeling.

Definition 2 Let $(V,\mathcal{C})$ be a specification of a single repeating event I, where $\mathcal{C}$ contains a constraint $l_n \leq n_I \leq u_n$. Replace the variables d_I and g_I in V with a set of $2u_n - 1$ variables, $\{d_1, d_2, \ldots d_{u_n}, g_1, g_2, \ldots, g_{u_n-1}\}$. The resulting set, V_S, thus contains

$$V_S = \{n_I, d_1, d_2, \ldots d_{u_n}, g_1, g_2, \ldots, g_{u_n-1}, e_I\}.$$

A *complete atomic labeling for a repeating event* is a tuple of size $2u_n + 1$ of assignments to elements in V_S; it has the following form:

$$\langle val(n_I), val(d_1), \ldots val(d_{u_n}), val(g_1),$$

$$\ldots, val(g_{u_n-1}), val(e_I)\rangle$$

where $val(n_I)$ is a number in the range $[l_n, u_n]$, and the first $val(n_I)$ of the $val(d_j)$ are in the range $[l_d, u_d]$, and the remaining $val(d_j)$ are 0. Similarly,

the first $val(n_I) - 1$ of the $val(g_j)$ are in the range $[l_g, u_g]$, and the remaining $val(g_j)$ take the value 0. Finally, $val(e_I)$ is in the range $[l_e, u_e]$.

Example 2 Given a specification for a repeating event in the previous example, one complete atomic labeling for I is

$$\langle 4, 6, 6, 6, 6, 0, 0, 2, 2, 3, 0, 0, 31 \rangle$$

and states that the repeating event has 4 occurrences, with each subinterval duration of 6 time units and 3 gaps with 2, 2 and 3 time units, for a total extent of $31 = 6+6+6+6+0+0+2+2+3+0+0$ time units.

Complete atomic labelings serve as inputs to the optimization phase of the reasoning process, to which we now turn.

4 Optimization in reasoning about repeating events

As noted at the outset, some problems involving repeating events require optimization on one or more of the structural aspects. For example, it is common to formulate problems in which *as many (few) events are scheduled as possible* during a specific time frame, or that the extent of a repeating event be minimized. In the telescope scheduling problem described at the outset, an optimizing constraint on the *distribution* of gaps between sub-intervals of repeating events was specified. A representation of this problem requires gap durations between repeating observations to satisfy a certain pattern, such as a uniform distribution of values from a range. In this section, a formulation of these types of problems is presented using the notion of semiring based CSP (SCSP) [Bistarelli *et al.*, 1995]. This section follows the terminology found in that paper.

First, we describe a simple extension to the language for specifying the profile of single repeating events introduced above. The extension involves a set of functional operators $min(), max(), Unif(), \ldots$ which form what will be called *optimizing constraints*, i.e., constraints that have an optimizing condition associated with them. For example, $min(3 \leq n_I \leq 6)$ will state that the number of subintervals is constrained to be in the interval $[3, 6]$, and should be as few as possible. Similarly, drawing on the telescope example, we let $Unif(3 \leq g_I \leq 10)$ state that the gap durations are to be between 3 and 10 time units, and should be selected uniformly from this interval.

Example 3 Consider the following set of constraints on I:

- number: $Max(2 \leq n_I \leq 6)$
- sub-interval duration: $4 \leq d_I \leq 7$
- gap duration: $Unif(4 \leq g_I \leq 9)$
- extent: $Min(30 \leq e_I \leq 100)$

This example formalizes a set of constraints, including optimizing constraints on the number, gaps, and extent of the repeating event I. Preferred solutions will maximize the number of occurrences of I, subject to a uniform distribution of gaps, and a minimal total extent.

To solve reasoning problems about repeating events with optimizing constraints, a computational mechanism for evaluating complete atomic labelings, based on their adherence to the optimizing conditions on constraints, is required. As a first step in formalizing this process, it is useful to apply the notion of *projection* (Π) to retrieve parts of complete atomic labelings that can be evaluated. Thus, given

$$t = \langle val(n_I), val(d_1), val(d_2), \ldots, val(d_n),$$
$$val(g_1), val(g_2), \ldots, val(g_{n-1}), val(e_I) \rangle$$

we define:

$$\Pi_{n_I}(t) = \langle val(n_I) \rangle$$
$$\Pi_{d_I}(t) = \langle val(d_1), val(d_2), \ldots, val(d_n) \rangle$$
$$\Pi_{g_I}(t) = \langle val(g_1), val(g_2), \ldots, val(g_{n-1}) \rangle$$
$$\Pi_{e_I}(t) = \langle val(e_I) \rangle$$

We next define a set of evaluation functions and a simple semiring for evaluating consistent complete labelings.

Definition 3 A *c-semiring* is a tuple $(A, +, \times, \mathbf{0}, \mathbf{1})$, where A is a set and $\mathbf{0}, \mathbf{1} \in A$; $+$ is a closed, idempotent, communitive and associative operation, and $\mathbf{0}$ is its unit element; $\times$ is closed, commutative and associative and $\mathbf{1}$ is its unit element, $\times$ distributes over $+$, and $\mathbf{1}$ is the absorbing element of $+$.

Example 4 Let $S = (A, +, \times, \mathbf{0}, \mathbf{1})$ be a c-semiring for evaluating atomic labelings. The domain A contains four values $\{1, A_1, A_0, 0\}$ with the total ordering $1 \leq_S A_1 \leq_S A_0 \leq_S 0$. Intuitively, $\leq_S$ can be interpreted as the relation "better than" for tuples of values. Thus, 1 and 0 are assigned to the best and worst labelings, respectively, and the intermediate values A_1 and A_0 can be assigned the interpretation "pretty good" and "pretty bad" respectively. The functions $+$ and $\times$ defined on elements of A can be interpreted as returning the "min" and "max" of elements in A, respectively, based on $\leq_S$.

Formally, an optimizing constraint is a pair $\langle f, v \rangle$, where $f : D^k \rightarrow A$ is a function from tuples of values from the domains of variables in the set $v \subseteq V_S$, where $|v| = k$. In particular, based on the previous example, we define three constraints:

$$Max\_n = \langle Max, \{n_I\} \rangle$$
$$Min\_e = \langle Min, \{e_I\} \rangle$$

and

$$Unif\_g = \langle Unif, \{g_1, g_2, \ldots g_{u_n}\} \rangle.$$

These are functions for minimality, maximality, and uniformity in the values assigned to the relevant variables in V_S. For example, given the variable e_I for

extent and a domain constraint $l_e \leq e_I \leq u_e$, the function Min associated with the optimizing constraint Min_e will be defined as follows:

$$Min(\langle val \rangle) = \begin{cases} 1 & \text{if } val = l_e \\ 0 & \text{if } val = u_e \\ A_1 & \text{if } val \in [l_e + 1, \lceil \frac{u_e}{2} \rceil] \\ A_0 & \text{otherwise} \end{cases}$$

The definition of the optimizing constraint $Unif\_g$ is made slightly more complicated by the fact that it is defined over tuples of variables, some of which may be 0 based on the value assigned to n_I. Intuitively, a tuple t of duration assignments to gaps will be preferred by $Unif\_g$ to the extent the values in t are distinct. Of course, the number of distinct values depends both on the number of gaps that occur, and on the number of distinct domain elements specified by the constraint. We can define equivalence classes for each tuple of assignments to gap attributes based on the number of pairs of distinct values in t, and then assign each equivalent class to one of the 4 elements of A.

Formally, let $t_g = \Pi_{g_I}(t_s)$ be an n-tuple of values of gap variables from a complete labeling t_s. Let $val_i(t_g)$ be the ith value of t_g. We next define

$$Diff(t_g) = SizeOf(\{\langle val_i(t_g), val_j(t_g) \rangle : val_i(t_g) \neq val_j(t_g), 1 \leq i < j \leq n\}).$$

$Diff$ is the number of distinct pairs of values in t_g. For each pair of tuples t_g and t'_g, define $t_g \equiv t'_g$ if $Diff(t_g) = Diff(t'_g)$. The result is a set of equivalence classes $\mathcal{D}$ based on $\equiv$. We "normalize" this set by mapping the p elements of $\mathcal{D}$ to the first p integers, such that the largest value of $Diff(t_g)$ is assigned 1, the second is assigned 2, etc. Let $norm(Diff(t))$ refer to the integer assigned to $Diff(t)$ in the normalization. We can then define

$$Unif(t_g) = \begin{cases} 0 & \text{if } norm(Diff(t_g)) = 1 \\ 1 & \text{if } norm(Diff(t_g)) = p \\ A_0 & \text{if } 1 < norm(Diff(t_g)) \\ & \quad \leq \lceil \frac{p}{2} \rceil \\ A_1 & \text{otherwise} \end{cases}$$

Thus, $Unif$ prefers tuples with more distinct pairs of elements, as intended.

Using the multiplication operation defined on semiring domains, each of the projections to tuples that correspond to optimization constraints can be combined to evaluate a complete labelings.

Example 5 Revisiting Example 3, and using the c-semiring of Example 4, consider the complete labeling

$$t = \langle 4, 4, 6, 6, 5, 0, 0, 4, 3, 4, 0, 0, 32 \rangle.$$

The projections to each of the optimizing constraints are the following:

- $\Pi_{n_I}(t) = \langle 4 \rangle$
- $\Pi_{g_I}(t) = \langle 4, 3, 4, 0, 0 \rangle$
- $\Pi_{e_I}(t) = \langle 32 \rangle$

Furthermore, $Diff(\Pi_{g_I}(t)) = 8$, and there are 5 equivalence classes defined by this operation for tuples of this size. The reader can verify that $norm(Diff(\Pi_{g_I}(t))) = 2$, and therefore $Unif(\Pi_{g_I}(t)) = A_1$. Furthermore, $Max(\Pi_{n_I}(t)) = A_1$, and $Min(\Pi_{e_I}(t)) = A_0$. Thus, t is assigned the evaluation $A_1 \times A_1 \times A_0 = A_0$.

This simple example was intended merely to illustrate the process of temporal optimization. A more robust evaluation would be based on a c-semiring with a larger domain, thus allowing more distinct values to be associated with complete labelings, providing greater granularity in the evaluation.

With the mechanism in place for evaluating complete atomic labelings, it is possible to generate and evaluate profiles of single repeating events based on specifications containing optimizing constraints. Space limitations prohibit an extensive discussion of the complete search process; a brief informal summary suffices here. The space of complete atomic labelings provides the input to what amounts to a constrained optimization problem. Local search algorithms have been successfully applied to such problems, and it is quite easy to see how they could be applied here. For example, a simple "greedy" approach would use concretizations to iteratively generate local modifications to consistent labelings. Changes that result in better evaluations are saved, and the process repeated for a fixed number of iterations.

5 Extensions to binary constraints

The framework discussed here allows for temporal constraints, including optimization constraints, on single repeating events to be specified. To be applicable to real scheduling problems, the framework must be extended to represent binary relations between distinct repeating events. Representing binary relations on repeating events has been the subject of previous work by the same authors, and recent results are the subject of a companion paper to this one (citation omitted to preserve anonymity), which should be consulted for more details. In this section, we give an brief presentation of this extension.

This paper has shown that by quantifying over start- or end-points of subintervals of a single repeating event, diagonal constraints can be specified. Generalizing, by comparing one endpoint of I with one of J, information about the relative ordering of these repeating events can be extracted, resulting in a language for specifying binary diagonal constraints. For example, a constraint such as that *the start of each sub-interval of J should be between 2 and 4 times units after the start of some occurrence of I* can be formulated in first-order logic as the period constraint

$$\forall J_i \in J \; \exists I_k \in I \; s(J_i) - s(I_k) \in [2, 4].$$

Second, the notion of concretization introduced in this paper can be extended to binary relations. A concretization of a binary constraint between pairs of repeating events consists of a mapping between start- or end-points of sub-intervals of I and those of sub-intervals of J. Different mappings are possible, depending on things like the number of each repeating event, and the type of mapping implied (e.g., one-to-one, onto). This mapping becomes the trigger to the relation concretization. The method described in this paper for profiling the diagonal of a single repeating event can be extended to profiling a set of distinct repeating events with binary constraints specified between them. Future work will be aimed at integrating the results of this paper with the generalized specification language for binary relations between repeating events.

6 Related work

The computational aspects of periodicity have been studied in the database literature, in specifying and verifying the correctness of continuously operating concurrent programs such as operating systems and, to a somewhat lesser extent, in AI. Modal temporal logic provides the framework for much of this research. Classical temporal logic does not allow for most forms of periodic knowledge to be expressed. To attain the needed expressive power, fixed point operators must be introduced; some fixed point extensions (e.g., Vardi's USF logic) to propositional modal logic are decidable. Alternatively, second-order extensions to predicate logic have been proposed to formalize the logic of infinite, including repeating, sequences; the system S1S (monadic second-order theory of successor) is decidable, and has the advantage of possessing a decision procedure in the form of Büchi automata.

In the AI literature, the interest in repeating events has stemmed from the desire to represent calendar time and for specifying and querying planning or scheduling information. The work reported here, as suggested by the example at the outset, is motivated primarily by scheduling problems. The approach of clustering a set of intervals and imposing a temporal structure on the cluster in order to reduce the burden of the temporal reasoner is reminiscent of the notion of a reference interval, first introduced in [Allen, 1983].

7 Summary

This paper has presented a formulation of repeating events within the CSP framework. What distinguishes such reasoning problems from temporal reasoning about single events is the need to manipulate information about the number of times an event can occur, and the distribution of those occurrences over time. This leads to a language and framework for reasoning in which constraints are formulated about the number and period of each repeating event. This framework exploits the temporal structure underlying repeating events to simplify the reasoning process, and a simple extension allows for optimization of diagonal profiles to be performed.

References

[Allen, 1983] James Allen. Maintaining Knowledge About Temporal Intervals. In *Readings in Knowledge Representation*, Brachman, R., and Levesque, H., pages 510-521, Morgan Kaufmann Publishers.

[Bistarelli *et al.*, 1995] Stefano Bistarelli, Ugo Montanari, and Francesca Rossi. Constraint Solving over Semirings. In *Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence*, pages 624-630, Morgan Kaufmann Publishers.

[Bresina, 1994] John Bresina. Telescope Loading: A Problem Reduction Approach. In *Proceedings of the Third International Symposium on Artificial Intelligence, Robotics, and Automation for Space.* Pasadena, CA, Jet Propulsion Labs.

[Dechter *et al.*, 1991] Rina Dechter, Itay Meiri, and Judea Pearl. Temporal Constraint Networks. *Artificial Intelligence*, 49:61-95, 1991.

[Morris *et al.*, 1996] Robert Morris, William Shoaff and Lina Khatib. Domain Independent Reasoning about Recurring Events. *Computational Intelligence*, 12(3):450-477, 1996.