Online appendix for the paper *Finding Optimal Plans for Multiple Teams of Robots through a Mediator: A Logic-Based Approach* published in Theory and Practice of Logic Programming

Esra Erdem, Volkan Patoglu, Zeynep G. Saribatur, Peter Schüller
Faculty of Engineering and Natural Sciences, Sabancı University, İstanbul, Turkey
{esraerdem,vpatoglu,zgsaribatur,peterschueller}@sabanciuniv.edu

Tansel Uras
Department of Computer Science, University of Southern California, Los Angeles, USA
turas@usc.edu

submitted 10 April 2013; revised 23 May 2013; accepted 23 June 2013

Appendix A Additional Experiments

A.1 Answering Queries: ASP vs. SAT

We can answer queries using CCalc with a SAT solver or with an ASP solver. In the former case, given an action description and a query, CCalc finds an answer to the query in the spirit of satisfiability planning (Kautz and Selman 1992): 1) it transforms the action description and the query into a set of formulas in propositional logic (Giunchiglia et al. 2004); 2) it calls a SAT solver (like manysat) to find a model of all these formulas; 3) if a model is found then it extracts the solution; otherwise, it answers the query negatively.

In the latter case, given an action description and a query in the language of CCalc, 1) we can use cplus2asp (Casolary and Lee 2011) to transform the action description and the query into an ASP program; 2) an ASP solver (like Clasp with the grounder

Table A 1. *Experimental results comparing ASP vs. SAT*

Scenario	Teams	Workspace (grid cells)	Worker Robots	Total Robots	Order (boxes)	Questions (total)	Answering Questions (average time ASP)	Answering Questions (average time SAT)
	#	#	#	#	#	#	sec	sec
1	2	15	1,2	5	6	212	3.96	6.67
2	3	15	1,2,3	9	9	437	3.92	6.13
3	4	15	1,2,3,4	15	12	525	1.82	3.36
4	2	24	2,4	8	8	127	4.76	7.91
5	3	24	2,4,6	18	12	171	5.37	13.08
6	4	24	2,4,6,8	30	16	293	79.96	151.33

Table A 2. *Experimental results comparing two approaches to compute optimal values for plan length $\bar{l}$: linear search vs. binary search*

Search Method	Scenario	Teams	Workspace (grid cells)	Worker Robots	Total Robots	Order (boxes)	Questions (total)	Answering Questions (average time)
		#	#	#	#	#	#	sec
$\bar{l}$: linear	1	2	15	1,2	5	6	212	6.67
	2	3	15	1,2,3	9	9	437	6.13
	3	4	15	1,2,3,4	15	12	525	3.36
	4	2	24	2,4	8	8	127	7.91
	5	3	24	2,4,6	18	12	171	13.08
	6	4	24	2,4,6,8	30	16	293	151.33
$\bar{l}$: binary	1	2	15	1,2	5	6	100	8.78
	2	3	15	1,2,3	9	9	187	9.96
	3	4	15	1,2,3,4	15	12	310	7.78
	4	2	24	2,4	8	8	163	215.13
	5	3	24	2,4,6	18	12	201	224.32
	6	4	24	2,4,6,8	30	16	351	283.63

GRINGO) can be used to compute an answer set for the program; 3) if an answer set is found then we can extract the solution; otherwise, the query is answered negatively.

We performed experiments to compare these two approaches, with the same instances used in our experiments, as explained in Section 6. Table A 1 summarizes the results of these experiments, comparing the computation times using the ASP solver CLASP with the grounder GRINGO, with the computation times using CCALC with the multi-threaded SAT-solver MANYSAT (limited to four threads). The computation times are average CPU times in seconds, obtained over three repeated runs of all scenarios. The time reported for ASP includes the time spent for grounding by GRINGO; the time reported for SAT includes the time spent for obtaining the propositional theory by CCALC. Note that except for the computation times used to answer queries, all other numbers are the same as in Table 1.

We observe from these results that the ASP solver CLASP performs better than CCALC with the SAT solver MANYSAT in all cases. This is also true for real time (not shown in tables), not only for CPU time (shown in tables).

A.2 Finding Optimal Values: Linear vs. Binary Search

To find the optimal value for a global plan length $\bar{l}$, and to find the earliest/latest lend/borrow times l of individual teams, we can use binary search or linear search.

As discussed in Section 5, one possibility is to apply binary search between 1 and $\bar{l}$ to find the earliest lend times and the latest borrow times l, and between 1 and k to find the optimal value for the global plan length $\bar{l}$. With this approach, every team answers $O\big(\overline{m}{\cdot}log(k)^2\big)$ queries.

However, the computation time to answer a query drastically increases as the plan length increases (due to inherent hardness of planning (Turner 2002; Erol et al. 1995)). In such cases, as suggested by Trejo et al. (Trejo et al. 2001), it is not a good idea to apply binary search to find the optimal value for a global plan length $\bar{l}$.

Meanwhile, given a plan length, queries to find the earliest lend times and the latest borrow times take about the same time; in such cases, as also suggested by Trejo et al., it is a good idea to apply binary search to find these optimal values.

Therefore, a better approach might be to use linear search to find the optimal value for a global plan length $\bar{l}$, and binary search to find optimal values for lending/borrowing times.

We compared these two approaches experimentally over the six scenarios used in our experiments (Section 6), using CCalc with manysat. Table A 2 shows the results of these experiments. Results are averages over three runs.

The first section of the table shows the configuration which was already used in Table 1 and Table A 1: $\bar{l}$ is determined using linear search and within teams the earliest lend and latest borrow times are determined using binary search. The second section of Table A 2 shows the strategy using binary search in both cases.

We can observe that these experimental results confirm Trejo et al. (Trejo et al. 2001)'s results summarized above. For small scenarios, the overall time to find a solution (not shown in the table) is smaller with two binary searches while for large scenarios using linear search for $\bar{l}$ gives a better overall performance. For example, for Scenario 1, a total of 1012 seconds is required to find the optimal value for $\bar{l}$ with binary search, while using linear search requires 1670 seconds. On the other hand, finding the optimal value for $\bar{l}$ in Scenario 4 requires 36737 seconds using binary search while linear search requires only 2114 seconds.

Nevertheless the overall time to find the optimal solution increases in all scenarios either due to an increased effort for answering questions or due to a significantly increased amount of queries.

Appendix B Proofs

Proof of Proposition 1
Let us denote by FindCollaboration$_D$ the decision version of FindCollaboration, i.e., decide for an existence of a $\overline{ml}$-collaboration. We prove that FindCollaboration$_D$ is NP-complete in two parts: membership and hardness.

Membership: Let $\Sigma = \{0, 1, \wedge, \vee, \cdot, \circ, \bullet, \star\}$ be the alphabet, and let Σ^* denote the set of all strings over Σ^*. We define a language L to be the set of all strings in Σ of the form $B_1 \wedge B_2 \wedge B_3 \wedge B_4 \wedge D \wedge X \wedge Y$ where

- B_1, B_2, B_3, B_4 are binary representations of the number of Lenders $|\mathit{Lenders}|$, the number of Borrowers $|\mathit{Borrowers}|$, the maximum number of steps $\bar{l}$, and the maximum number of robots $\overline{m}$, respectively.
- D has the form $D_{1,1} \wedge D_{1,2} \wedge ... \wedge D_{a,b}$ with each $D_{i,j}$ of the form $(I_d \wedge J_d) \cdot D'$ where I_d and J_d are the binary representation of lender index i and borrower index j, and D' is the binary representation of the value $\mathit{Delay}(i, j)$.
- X has the form $X_{m_1,i_1} \wedge X_{m_1,i_2} \wedge \cdots$ with each $X_{m,i}$ of the form $(M_x \wedge I_x) \circ S_x$

where M_x and I_x are the binary representation of number m and lender index i, and S_x is the binary representation of the value $\mathit{Lend\_earliest}_m(i)$.
- Y has the form $Y_{m_1,j_1} \wedge Y_{m_1,j_2} \wedge \cdots$ with each $Y_{m,j}$ of the form $(M_y \wedge J_y) \bullet S_y$ where M_y and J_y are the binary representation of number m and borrower index j, and S_y is the binary representation of the value $\mathit{Borrow\_latest}_m(j)$,

such that, given an input $x \in \Sigma^*$ then $x \in L$ iff FindCollaboration$_D$ with input corresponding to x returns yes.

Note that FindCollaboration$_D$ is a decision problem since FindCollaboration$_D$ returns yes if and only if $x \in L$.

We will show that FindCollaboration$_D$ is in NP by showing that (A) the above representation $x \in \Sigma^*$ is polynomial in the size of an input to FindCollaboration$_D$, (B) we can describe a guess y of polynomial size corresponding to a potential collaboration function f, and (C) checking whether y satisfies all conditions of Def. 1 with respect to input x can be done by a polynomial time algorithm.

(A) The input $x \in \Sigma^*$ consists of four numbers and at most $1{+}2{\cdot}\overline{m}$ functions (*Delta* plus a maximum of $\overline{m}$ *Lend_earliest* and *Borrow_latest* functions), where *Delta* has size $\mathcal{O}(|\mathit{Lenders}| \cdot |\mathit{Borrowers}| \cdot \log(\overline{l}))$ and the other functions are below that. Therefore, $|x|$ has polynomial size in $\overline{l}{\cdot}\overline{m}$.

(B) A guess y, corresponding to a collaboration function f, will be of the form $F_1 \wedge \cdots \wedge F_w$ with $w = \mathcal{O}(|\mathit{Lenders}| \cdot |\mathit{Borrowers}| \cdot \overline{l} \cdot \log(\overline{m})) = \mathcal{O}(|x|^4)$ where each F_i is of the form $(I_u \wedge J_u) \star (L_u \wedge M_u)$ with I_u, J_u, L_u, M_u the binary representations of lender i_u, borrower j_u, time step l_u and number of robots m_u, for $f(i_u, j_u) = (l_u, m_u)$. Each F_i has linear size in $|x|$ therefore $|y| = \mathcal{O}(|x|^5)$ and therefore polynomial.

(C) We can check whether y conforms to all conditions in polynomial time: For condition (a) and (b), we check at most $\overline{m}$ values of *Borrow_latest* and *Lend_earliest* functions for each *Borrower* and *Lender*, respectively. Therefore, the checking algorithm takes polynomial time in $|x|$.

Hence, FindCollaboration$_D$ is in NP.

Hardness: Take any 3-SAT instance F over signature σ of n variables $x_1, \ldots, x_n$ and p clauses $c_1, ..., c_p$ of the form $c_j = (t_{j,1}, t_{j,2}, t_{j,3})$, where $t_{j,1}, t_{j,2}, t_{j,3}$ are literals. We can reduce F to an instance of FindCollaboration$_D$ as follows.

First, we define the sets *Lenders* and *Borrowers*. The set of *L*enders has n lenders for each variable in F. We define a function $\varphi : \mathit{Lenders} \to \sigma$ such that $\varphi(i) = x_i$, to denote the relation between the lenders and the variables, $\mathit{Lenders} = \{1, ..., n\}$. The set of *B*orrowers is defined for each clause in F, $\mathit{Borrowers} = \{n{+}1, ..., n{+}p\}$. So there is a 1-1 mapping between lenders and variables, and between borrowers and clauses.

We define $\overline{l} = 2 * |\sigma| = 2n$, and a mapping $\mathit{step}(x_i)$ from literal x_i, (resp., $\neg x_i$) to time steps such that $\mathit{step}(x_i) = i$, (resp., $\mathit{step}(\neg x_i) = n + i$).

Given a literal t in F, we denote by $\mathit{occ}(t)$ the number of clauses of F which contain t. Without loss of generality, we assume that no literal is contained twice in any clause. Using $\mathit{occ}(t)$ we define a function $\mathit{rNum} : \{1, \ldots, \overline{l}\} \to \{1, \ldots, \overline{m}\}$ where $\overline{m}$ is a positive integer explained below; *rNum* associates each time step (i.e., each literal) with a number of robots.

Intuitively, for each clause $c \in F$ containing literal t, $\mathit{rNum}(\mathit{step}(t))$ robots must be transferred to satisfy c. To achieve this, we define rNum such that for each time step u, the number of robots that must be transferred is larger than the total number of robots that can be transferred before u, i.e., larger than the number of robots in all previous steps multiplied by their respective occurrence counts of associated literals:

$$\begin{aligned}
&\mathit{rNum}(1) = 1\\
&\mathit{rNum}(u) = 1 + \textstyle\sum_{i=1}^{u-1} \mathit{rNum}(i) \cdot \mathit{occ}(\mathit{step}^{-1}(i)) \quad \text{for} \quad 1 < u \leq \bar{l}
\end{aligned}$$

The constant $\overline{m}$ is the maximum number of robots that can be given at the latest time step, i.e., the largest value of rNum for a given 3-SAT instance. This value is $\overline{m} = \mathit{rNum}(\mathit{step}(\neg x_n)) \cdot \mathit{occ}(\neg x_n)$; by eliminating the definition of rNum from the formula, we obtain the following equation

$$\overline{m} = \big(\mathit{occ}(x_1)+1\big) \cdot \ldots \cdot \big(\mathit{occ}(x_n)+1\big) \cdot \big(\mathit{occ}(\neg x_1)+1\big) \cdot \ldots \cdot \big(\mathit{occ}(\neg x_{n-1})+1\big) \cdot \mathit{occ}(\neg x_n).$$

Since a literal may occur in at most p clauses, $\overline{m} = \mathcal{O}(p^{2n})$ which is exponential in the input size. This is not a problem as the value $\overline{m}$ can be computable in polynomial time and represented in linear space, moreover our reduction never requires to explicitly represent $\overline{m}$ functions: *Lend_earliest* is defined on three intervals per lender and *Borrow_latest* is defined on four intervals per borrower, hence a polynomial time reduction.

We define $\mathit{Delay}(i,j) = 0$, for every $i \in \mathit{Lenders}, j \in \mathit{Borrowers}$.

For each lender i, we define $\mathit{Lend\_earliest}_m(i)$ as follows:

$$\mathit{Lend\_earliest}_m(i) = \begin{cases} \mathit{step}(\varphi(i)) & \text{if } 1 \leq m \leq \mathit{rNum}(\mathit{step}(\varphi(i))) \cdot \mathit{occ}(\varphi(i)) \\ \mathit{step}(\neg\varphi(i)) & \text{if } \mathit{rNum}(\mathit{step}(\varphi(i))) \cdot \mathit{occ}(\varphi(i)) < m \text{ and} \\ & \quad m \leq \mathit{rNum}(\mathit{step}(\neg\varphi(i))) \cdot \mathit{occ}(\neg\varphi(i)) \\ \text{undefined} & \text{otherwise} \end{cases}$$

Intuitively, each lender i can lend up to $\mathit{rNum}(\mathit{step}(\varphi(i))) \cdot \mathit{occ}(\varphi(i))$ robots with earliest time step $\mathit{step}(\varphi(i))$ or it can lend up to $\mathit{rNum}(\mathit{step}(\neg\varphi(i))) \cdot \mathit{occ}(\neg\varphi(i))$ robots with earliest time step $\mathit{step}(\neg\varphi(i))$.

For each clause $c_j = (t_{j,1}, t_{j,2}, t_{j,3})$ in F, without loss of generality, we assume that $\mathit{step}(t_{j,1}) \leq \mathit{step}(t_{j,2}) \leq \mathit{step}(t_{j,3})$, and, for each borrower $j{+}n$, we define $\mathit{Borrow\_latest}_m(j{+}n)$ as follows:

$$\mathit{Borrow\_latest}_m(j{+}n) = \begin{cases} \text{undefined} & \text{if} \quad 1 \leq m < \mathit{rNum}(\mathit{step}(t_{j,1})) \\ \mathit{step}(t_{j,1}) & \text{if} \quad \mathit{rNum}(\mathit{step}(t_{j,1})) \leq m < \mathit{rNum}(\mathit{step}(t_{j,2})) \\ \mathit{step}(t_{j,2}) & \text{if} \quad \mathit{rNum}(\mathit{step}(t_{j,2})) \leq m < \mathit{rNum}(\mathit{step}(t_{j,3})) \\ \mathit{step}(t_{j,3}) & \text{if} \quad \mathit{rNum}(\mathit{step}(t_{j,3})) \leq m \leq \overline{m} \end{cases}$$

Intuitively, each borrower $j{+}n$ corresponding to clause c_j needs to borrow at least the number of robots associated with at least one literals in c_j, at the latest time step that is associated with that particular literal.

Example 1 *Figure B 1 shows an example reduction from 3-SAT formula* $F_1 = (a \vee b \vee \neg c) \wedge (c \vee \neg a \vee \neg b)$. *Bold lines indicate Lend_earliest and Borrow_latest functions, e.g., lender 2, corresponding to variable b has* $\mathit{Lend\_earliest}_{16}(2) = 5$. *Numbers given next to bold lines indicate values of rNum for the respective step, e.g.,* $\mathit{rNum}(5) = 16$. *Note that* $\mathit{occ}(t) = 1$ *for every literal t, hence* $\mathit{rNum}(\mathit{step}(t)) = 2^{\mathit{step}(t)}$.

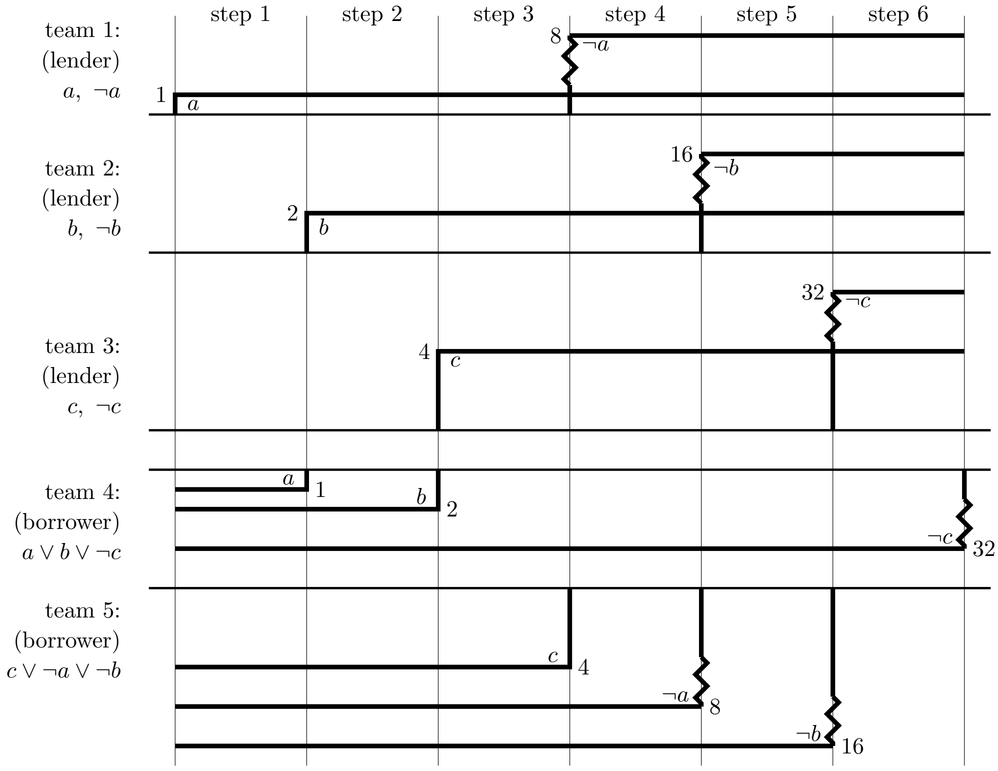

Fig. B 1. Example hardness reduction for 3-SAT formula $F_1 = (a \vee b \vee \neg c) \wedge (c \vee \neg a \vee \neg b)$

Note that the reduction from 3-SAT to FindCollaboration$_D$ can be done in time polynomial in the size of the input formula. Let us prove that this is a correct reduction: F is satisfiable iff there is an $\overline{ml}$-collaboration between *Lenders* and *Borrowers* with at most $\overline{m}$ robot transfers and at most in $\bar{l}$ steps defined above.

Hardness: SAT $\rightarrow$ collaboration Let I be an interpretation mapping σ to truth values such that this assignment satisfies F. Here and in the following we denote an interpretation I by the set of atoms in σ whose values are mapped to true.

We define the collaboration function

$$f : Lenders \times Borrowers \rightarrow \{0, ..., \bar{l}\} \times \{0, ..., \overline{m}\}$$

as follows:

- for every variable $s \in I$ and for every borrower $j{+}n$,

  $$f(\varphi^{-1}(s), j{+}n) = (step(s), rNum(step(s)))$$

  where clause c_j contains s;
- for every variable $s \notin I$ and for every borrower $j + n$,

  $$f(\varphi^{-1}(s), j{+}n) = (step(\neg s), rNum(step(\neg s)))$$

  where clause c_j contains $\neg s$.

Example 2 (ctd) *Interpretation $I_1 = \{a, c\}$ satisfies F_1 and induces the following collaboration f_1: $f_1(1,4) = (1,1)$, $f_1(2,5) = (5,16)$, $f_1(3,5) = (3,4)$.*

We can now show that f, as obtained above from I, indeed satisfies all conditions of Def. 1, i.e., it is an $\overline{ml}$-collaboration.

- Def. 1(a): I satisfies each clause in F. For every borrower j corresponding to clause $c_j = (t_{j,1}, t_{j,2}, t_{j,3})$, let $t_{j,k}$ be a literal in c_j satisfied by I. Take any borrower $j{+}n$. By our construction of f, there is a lender $i_k = \varphi^{-1}(var(t_{j,k}))$ such that $f(i_k, j+n) = (step(t_{j,k}), rNum(step(t_{j,k})))$ where $step(t_{j,k}) \leq \bar{l}$ and $rNum(step(t_{j,k})) \leq \overline{m}$. Take $m = rNum(step(t_{j,k})) \leq \overline{m}$. Then the following hold:

$$\begin{aligned} \max\{step(t_{j,k})\} \leq step(t_{j,k}) &= Borrow\_latest_m(j{+}n) \\ m &\leq rNum(step(t_{j,k})). \end{aligned}$$

  Hence condition (a) holds.
- Def. 1(b): Take any lender i (corresponding to variable $\varphi(i)$). Consider two cases:

— Case 1: $\varphi(i) \in I$. Lender i has cooperations with borrowers $n{+}j_1, .., n{+}j_q$, corresponding to clauses $c_{j_1}, ..., c_{j_q}$ which contain $\varphi(i)$. Note that $q \leq occ(\varphi(i))$ and $q = occ(\varphi(i))$ if and only if $\varphi(i)$ does not occur multiple times in any clause. The construction of f is as follows:

$$\begin{aligned} f(i, n{+}j_1) &= (step(\varphi(i)), rNum(step(\varphi(i)))) \\ &\vdots \\ f(i, n{+}j_q) &= (step(\varphi(i)), rNum(step(\varphi(i)))) \end{aligned}$$

where $step(\varphi(i)) = l_1 = \cdots = l_s \leq \bar{l}$, and $rNum(step(\varphi(i))) = u_1 = \cdots = u_s \leq \overline{m}$. Take $m = rNum(step(\varphi(i))) \cdot occ(\varphi(i)) \leq \overline{m}$. Then the following hold:

$$\begin{aligned} Lend\_earliest_m(i) = step(\varphi(i)) &\leq \min\{step(\varphi(i))\} \\ m &\geq rNum(step(\varphi(i))) \cdot q. \end{aligned}$$

— Case 2: $\varphi(i) \notin I$. Lender i has cooperations with borrowers $n{+}j_1, .., n{+}j_q$, corresponding to clauses $c_{j_1}, ..., c_{j_q}$ which contain $\neg\varphi(i)$. Note that $q = occ(\neg\varphi(i))$ and $q = occ(\neg\varphi(i))$ if and only if $\neg\varphi(i)$ does not occur multiple times in any clause. The construction of f is as follows:

$$\begin{aligned} f(i, n{+}j_1) &= (step(\neg\varphi(i)), rNum(step(\neg\varphi(i)))) \\ &\vdots \\ f(i, n{+}j_q) &= (step(\neg\varphi(i)), rNum(step(\neg\varphi(i)))) \end{aligned}$$

where $step(\neg\varphi(i)) = l_1 = \cdots = l_s \leq \bar{l}$ and $rNum(step(\neg\varphi(i))) = u_1 = \cdots = u_s \leq \overline{m}$. Take $m = rNum(step(\neg\varphi(i))) \cdot occ(\neg\varphi(i)) \leq \overline{m}$. Then the following hold:

$$\begin{aligned} Lend\_earliest_m(i) = step(\neg\varphi(i)) &\leq \min\{step(\neg\varphi(i))\} \\ m &\geq rNum(step(\neg\varphi(i))) \cdot q. \end{aligned}$$

Hence condition (b) holds.

Therefore, a function f obtained as shown above from a satisfying assignment I of F is a collaboration according to Def. 1.

Hardness: collaboration $\rightarrow$ SAT

Let f be a collaboration function defined via the reduction explained above. We need to show that there is an interpretation that satisfies F. Without loss of generality, we assume that $\mathit{Delay}(i,j)=0$ for all i,j, and do not mention delay in the following.

Since f is a collaboration function, it satisfies the conditions in Def. 1.

Given f and a borrower $j{+}n$ corresponding to clause c_j, we say that *borrower $j{+}n$ can complete its task with respect to a literal $t\in c_j$* iff the borrower borrows at least $\mathit{rNum}(\mathit{step}(t))$ robots up to time step $\mathit{step}(t)$ (inclusive) and there is no $t'\in c_j$ with $\mathit{step}(t)<\mathit{step}(t')$ such that the borrower borrows $\mathit{rNum}(\mathit{step}(t'))$ or more robots at a step after $\mathit{step}(t)$.

Intuitively, a borrower can complete its task with respect to literal $t\in c_j$ iff its task can be completed by obtaining robots until $\mathit{step}(t)$ and this is not true for another literal after that step. Given a collaboration function f, per definition of collaboration each borrower can complete its task with respect to *exactly one* literal.

Example 3 (ctd) *In collaboration f_1, borrower 4 can complete its task with respect to a and with respect to no other literal, borrower 5 can complete its task with respect to $\neg b$, and not with respect to c because $\mathit{step}(c)<\mathit{step}(\neg b)$.*

Towards proving the result, we introduce two lemmas.

Lemma 1 *If borrower $j{+}n$, corresponding to clause c_j, can complete its task with respect to literal $t_u\in c_j$, then borrower $j{+}n$ borrows at least one robot from lender $\varphi^{-1}(\mathit{var}(t_u))$ at step $\mathit{step}(t_u)$ and that robot cannot be lent before step $\mathit{step}(t_u)$.*

Intuitively, this lemma states that a borrower which can complete its task with respect to a certain literal t_u always borrows at least one robot from the lender corresponding to $\mathit{var}(t_u)$, and that robot is exchanged at the step corresponding to t_u, i.e., at the step corresponding to the correct (with respect to polarity of the variable) literal.

Example 4 (ctd) *Given f_1 we saw that borrower 5 can complete with respect to $\neg b$. According to Lemma 1, borrower 5 receives at least 1 robot at $\mathit{step}(\neg b)=5$ which is true as $f_1(2,5)=(5,16)$. Intuitively, Lemma 1 holds because borrower 5 requires 16 robots to complete its task with respect to $\neg b$ whereas lenders can only lend 15 robots in total during steps $1\ldots4$. Therefore, borrower 5 must receive at least 1 of the 16 robots that can be given at step 5, and this robot is in addition to 2 robots that could potentially be given at step 2. (See also Lemma 2 which shows that these 2 robots cannot be given in that case.)*

Proof of Lemma 1

- Case 1: t_u is a positive literal.
  Borrower $j{+}n$ can complete its task with respect to $t_u\in c_j$, so it borrows at least $\mathit{rNum}(\mathit{step}(t_u))$ robots with the latest time step $\mathit{step}(t_u)$.
  Borrower $j{+}n$ can cooperate with lenders $\varphi^{-1}(\mathit{var}(t_1)),\ldots,\varphi^{-1}(\mathit{var}(t_u))$.
  Towards a contradiction, assume that borrower $j{+}n$ does not borrow any robots from lender $\varphi^{-1}(\mathit{var}(t_u))$ at step $\mathit{step}(t_u)$. Then there exists a collaboration function

f with:

$$f(\varphi^{-1}(var(t_\alpha)), j{+}n) = (l_\alpha, m_\alpha)$$
$$\vdots$$
$$f(\varphi^{-1}(var(t_\beta)), j{+}n) = (l_\beta, m_\beta)$$

where $step(t_1) \leq l_\alpha \leq l_\beta < step(t_u)$ and $rNum(step(t_u)) \leq (m_\alpha + \cdots + m_\beta)$ and $\max\{l_\alpha, \ldots, l_\beta\} \leq step(t_u)$. Note that this function excludes borrowing at step $step(t_u)$ hence it excludes

$$f(\varphi^{-1}(var(t_u)), j{+}n) = (step(t_u), m) \text{ for every } m.$$

Therefore, the maximum number of robots that borrower $j{+}n$ can borrow with f is

$$rNum(step(t_1))\cdot occ(t_1) + \cdots + rNum(step(t_{u-1}))\cdot occ(t_{u-1}) =$$
$$= \sum_{i=1}^{u-1} rNum(step(t_i))\cdot occ(t_i) = rNum(step(t_u)) - 1$$

which is exactly one robot less than borrower $j{+}n$ needs to be able to complete with respect to t_u.

We have reached a contradiction: there cannot be a collaboration function that satisfies condition (a) of Def. 1 without lender $\varphi^{-}(var(t_u))$ lending at least one robot to borrower j at step $step(t_u)$.

- Case 2: t_u is a negative literal.

  If borrower $j{+}n$, corresponding to clause c_j, can complete its task with respect to literal $\neg t_u \in c_j$, then it borrows at least $rNum(step(\neg t_u))$ robots with the latest time step $step(\neg t_u)$.

  Assume that borrower $j{+}n$ borrows m robots from lender $\varphi^{-1}(var(t_u))$. Then,

  $$Lend\_earliest(\varphi^{-1}(var(t_u)))_m = step(t_u).$$

  By our construction of $Lend\_earliest$, we have $m \leq rNum(step(t_u))\cdot occ(t_u)$. With this assumption, there exists a collaboration function f with:

  $$f(\varphi^{-1}(var(t_\alpha)), j{+}n) = (l_\alpha, m_\alpha)$$
  $$\vdots$$
  $$f(\varphi^{-1}(var(t_\beta)), j{+}n) = (l_\beta, m_\beta)$$
  $$f(\varphi^{-1}(var(t_u)), j{+}n) = (l_\gamma, m)$$

  where $step(t_1) \leq l_\alpha \leq l_\beta \leq l_\gamma \leq step(\neg t)$ and $rNum(step(\neg t)) \leq (m_\alpha + \cdots + m_\beta + m)$ and $\max\{l_\alpha, \ldots, l_\beta, l_\gamma\} \leq step(\neg t)$.

  The maximum number of robots borrower $j{+}n$ can borrow can be computed as follows:

  $$rNum(step(t_1))\cdot occ(t_1) + \ldots + rNum(step(t_u))\cdot occ(t_u) + \ldots$$
  $$\ldots + rNum(step(\neg t_{u-1}))\cdot occ(\neg t_{u-1}) =$$
  $$= \sum_{i=1}^{u-1} rNum(step(t_i))\cdot occ(t_i) = rNum(step(\neg t_u)) - 1$$

  We have reached a contradiction: there cannot be a collaboration function that satisfies condition (a) of Def. 1 without lender $\varphi^{-}(var(t_u))$ lending at least one robot to borrower j at the time step where the lender has the earliest possibility to lend this robot, i.e., at step $step(t_u)$.

□

Lemma 2 *If borrower $j{+}n$, corresponding to clause c_j, can complete its task with respect to positive literal $t \in c_j$ (resp., with respect to negative literal $\neg t \in c_j$) then no borrower can complete its task with respect to negative literal $\neg t$ (resp., with respect to positive literal t).*

Intuitively, this lemma states that if a borrower can complete its task with respect to a certain literal t, no other borrower can complete its task with respect to the negation of literal t. In terms of truth values and clause satisfiability, Lemma 2 shows that a collaboration corresponds to a *consistent* set of literals satisfying all clauses.

Example 5 (ctd) *Given f_1 borrower 5 can complete with respect to $\neg b$, Lemma 2 states that no borrower can complete with respect to b. Intuitively, this holds because lender 2 provides at least one of 16 possible robots at step 5. Therefore, it cannot give robots already at step 2.*

Proof of Lemma 2

- Case 1: borrower $j{+}n$ can complete its task with respect to positive literal $t \in c_j$.
  By Lemma 1, borrower $j{+}n$ borrows at least one robot from lender $\varphi^{-1}(var(t))$ at step $step(t)$.
  Since lender $\varphi^{-1}(var(t))$ lends robots with earliest step $step(t)$, by *Lend_earliest* function, the lender can lend at most $rNum(step(t)){\cdot}occ(t)$ robots.
  Let borrower $j'{+}n$ be a borrower corresponding to a clause $c_{j'}$ which contains $\neg t$. To complete its task by borrowing robots from lender $\varphi^{-1}(var(t))$, it needs to borrow at least $rNum(step(\neg t))$ robots. However, $rNum(step(\neg t)){>}rNum(step(t)){\cdot}occ(t)$.
  Therefore, lender $\varphi^{-1}(var(t))$ cannot be the lender that satisfies borrower $j'{+}n$. In other words, clause $c_{j'}$ cannot be satisfied by literal $\neg t$.
- Case 2: borrower $j{+}n$ can complete with respect to negative literal $t \in c_j$.
  By Lemma 1, borrower $j'{+}n$ borrows $m \geq 1$ robots from lender $\varphi^{-1}(var(t))$ at step $step(\neg t)$, where
  $$Lend\_earliest(\varphi^{-1}(var(t))) = step(\neg t).$$
  Since lender $\varphi^{-1}(var(t))$ lends a number of robots with earliest time step as above, this lender cannot lend any robots before $step(\neg t)$. Therefore, lender $\varphi^{-1}(var(t))$ cannot be the lender that satisfies a borrower $j{+}n$ with respect to literal $t \in c_j$. In other words, clause c_j cannot be satisfied by literal t.

□

We can now prove that, if f is a $\overline{ml}$-collaboration then F is satisfiable.

As f is a collaboration function it satisfies the conditions in Def. 1. By condition (a) every borrower $j{+}n$ can complete its task with respect to some literal $t \in c_j$. Given f, let J be the set of all literals t s.t. some borrower can complete with respect to t. Due to Lemma 2 no two borrowers complete their respective tasks with respect to complementary literals x and $\neg x$ for some variable x. Hence J does not contain both positive and negative literals for any variable; it is a consistent set of literals.

Each borrower $j{+}n$ corresponding to clause c_j can complete its task with respect to some literal $t \in J$ and $t \in c$, therefore all clauses of F are satisfied by J. Therefore, given a collaboration function f, the consistent set of literals J corresponds to a (unique) satisfying truth assignment I for F. $\square$

References

Casolary, M. and Lee, J. 2011. Representing the language of the causal calculator in answer set programming. In *Proc. of ICLP (Technical Communications)*. 51–61.

Erol, K., Nau, D. S., and Subrahmanian, V. S. 1995. Complexity, decidability and undecidability results for domain-independent planning. *Artif. Intell. 76,* 1–2, 75–88.

Giunchiglia, E., Lee, J., Lifschitz, V., McCain, N., and Turner, H. 2004. Nonmonotonic causal theories. *AIJ 153*, 49–104.

Kautz, H. and Selman, B. 1992. Planning as satisfiability. In *Proc. of ECAI.* 359–363.

Trejo, R., Galloway, J., Sachar, C., Kreinovich, V., Baral, C., and Tuan, L.-C. 2001. From planning to searching for the shortest plan: An optimal transition. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems 9,* 6, 827–837.

Turner, H. 2002. Polynomial-length planning spans the polynomial hierarchy. In *Proc. of JELIA*. 111–124.