Distributed Algorithm for Incrementally Solving the Decoupled Multi-agent Simple Temporal Problem

Cu Nguyen Giap and Do Thi Thu Hien

***Abstract*—Applying temporal constraint in planning is a well-known problem, which keeps a plan is flexible until a specific schedule is generated. In this area, Decoupled Multi-agent simple temporal problem (DMaSTP) is suitably applied for planning of a multi-agents system. However, in scheduling problem, new events or temporal constraints are added regularly and force scheduler to check the consistency of exist MaSTP and retighten exist constraints. In this paper, we study a distributed scheduling algorithm for incrementally solving a DMaSTP. We have strongly considered the problem of adding a set of new constraints into a tightening consistent DMaSTP that tightened by also a distributed algorithm or set as empty. The algorithm checks whether the new adding constraints threaten the consistence of DMaSTP or not and decouple such new adding constraints when necessary and retighten DMaSTP. We have proposed a distributed algorithm, called DI-DMaSTP that solves the above problem and theoretically prove its correctness and outperformance, besides we have experienced with the variant datasets.**

***Index Terms*—Distributed algorithm, parallel, incremental, decoupled, multi-agents, simple temporal problem.**

I. Introduction

In a dynamic multi-agent system, an agent would have his private tasks and joint tasks, in which the tasks might be scheduled in different times and will be performed by an agent himself or by a set of agents. For example people might has a work plan involving shared tasks with his colleagues and he has private actions such as shopping. He would plan several joint events with his friends later also. These problems require a scheduler has to deal with new adding events all the time. In this circumstance Decoupled Muti-agent Simple Temporal Problem - DMa-STP - is a good candidate for planning multi-agent system at first [1].

In planning for an agent, the other important point besides scheduling at the first time is that the new events or temporal constraints are added regularly and force scheduler to retighten exist constraint and check the consistency. The problem of determining whether a tentative additional constraint will threaten a consistent DSTP is a simple question but retightening that DSTP is a NP-Hard problem [2]. Adding a set of additional constraints is an extension for adding a constraint and it is also a NP-Hard problem. We have determined that problem by the name Incremental Decoupling Multi-agent Simple Temporal Problem — IDMa-STP, and in this paper we have proposed a distributed algorithm to solve that problem, which has a largely potential application in reality.

There are several known approaches that could be taken to determine the consistency and to construct a decomposition of DMaSTP instance, from which a particular schedule is generated by a backtracking algorithm [2]. One is to gather all members' scheduling constraints, and solve the corresponding STP in a centralized fashion. Consequently, an incremental DMaSTP will be solved by repeatedly apply a single shoot central algorithm for STP or by applying a pure increment central algorithm [3]. The other approach is using a distributed algorithm for STP repeatedly.

In the centralized approach, Ma-STP is considered as a single STP and is solved incrementally by repeatedly applying single shoot algorithm, such as all pairs shortest path algorithms or an alternative algorithm Directed Path Consistency (DPC) [4], [5] or the L. Xu and B. Choueiry's algorithm, Δ STP, in [6] that extended from Partial Path Consistency (PPC) algorithm [7] and triangulated algorithm [8], [9]. The other useful algorithm applying PPC-based algorithms is represented by Planken, Weerdt, and Krogt [10] called new algorithm P3C, which sweep the triangles process in a systematic order, resulting in an improved performance. However, these algorithms require to recalculating all DMaSTP for every new adding constraint therefore they are so slow.

There are several pure incremental central algorithms proposed base on single shoot algorithms above. One is Incremental full path consistence – IFPC that based on full path consistence algorithm to check the consistence of STP and only update new bounds required by adding constraints. Besides, Planken has proposed an incremental algorithm for STP based on partial path consistence called IPPC [3]. However Planken's algorithm only works for adding constraints that do not threaten the triangulation property of the exist STP.

The disadvantage of all algorithms following the centralized fashion is that the scheduler has to gather agents' constraints, therefore any agent has to disclose its private tasks and this problem would not be welcomed if agents come from different organizations in reality.

In contradiction, a distributed algorithm gives an opportunity of maintaining the privacy of agents' private schedule and the distributed algorithm seeks out a suitable schedule of join event by exchanging required information but disclosing all agents' private schedules. J. Boerkoel has proposed a new formal definition of a STP for a scheduling problem of a Multi-agents system [1], [11]. In his point of view, an entire MaSTP has corresponding Decoupled MaSTP and his algorithm seeks for a Decoupled STP combining separated agents' schedules. As a DMa-STP is tightening, each private STP is used to generate an exact solution for each agent individually by a backtracking algorithm. He has

Manuscript received March 4, 2015; revised August 25, 2015.

The authors are with the Faculty of Economic Information System, Vietnam Commercial University, Vietnam (e-mail: cunguyengiap@vcu.edu.vn).

proposed an algorithm called D-MaTDP that improves not only computing time but also privacy of scheduling problem. However D-MaTDP has not solved the adding constraint problem.

In this study, we have strongly concerned the problem of adding a set of new constraints into a tight consistent DMa-STP, and how to check whether a new constraint threatens the consistence of such DMa-STP and how to decouple those constraints and retighten that DMa-STP. We have inherent J. Boerkoel definition of Ma-STP to solve DMa-STP incrementally. Even though, we have borrowed J. Boerkoel idea to decouple Ma-STP, our main contribution is dealing with a set of adding constraints into a tight DMa-STP.

We have proposed a distributed algorithm called DI-DMa-STP, in which each agent deals with local adding constraints privately and simultaneously by IFPC algorithms. Each agent also repeatedly decouples its adding inter-agent constraints by firstly setting decoupling constraints related to referent time point, and then retighten private STP by IFPC algorithms again. We have applied a decoupling strategy that maximizes the flexibility of all agents' private STPs. This approach increases the chance of successful planning when an agent attempts to make new a joint event.

In this paper, we have reviewed formal definitions related to DMa-STP problem in the second part and followed by details of our algorithm in the third part. We have also proved theoretically the soundness and completeness and upper bound of the algorithm's time complexity in the fourth part, and then compare our algorithm with the well-known existing algorithms practically in the fifth part. In the final part we summary several important points in our study and future works.

II. Background

In this part, we briefly review formal definition of STP, Multi-agents STP, Decoupled Multi-agents STP and several well-known solutions for such problems, which effected on our algorithm chronologically. We have also drawn the problem we have concerned in detail at the end.

The formal definition of a Simple temporal problem was first proposed in the paper "Temporal constraint network" by Dechter, Meiri and Pearl [5]. An important property of that definition is that a STP with all bilateral constraints can be represented by an undirected distance graph with all vertices and edges corresponding to time points and constraints of STP. When an STP is described by a constraint network, it is also call simple temporal network (STN). In the rest of this document we will use both pronouns STP and STN interchangeably.

A. Multi-agent Simple Temporal Problem

STP of a multi-agent system could be seen as one normal STP. Nonetheless, J. Boerkoel and H. Dufree have proposed a special definition of a STP of a multi-agent system as a set of private agents' STPs and a shared STP between all agents.

According to J. Boerkoel and H. Dufree [4], [11], the local STPs can be further partitioned into shared and private components. Particularly, V_A^i is partitioned into two sets, the agent i^{th}'s private time-points V_{AP}^i and the agent i^{th}'s shared time-points V_{AS}^i. The set edges of the agent i^{th}, E_N^i, is also partitioned into two sets, the set of private edges, E_{NP}^i, and shared inter-agent edges, E_{NS}^i. Note, shared inter-agent edges E_{NS}^i are edges whose endpoints are contained within the set V_{AS}^i.

The agent i$^{\text{th}}$'s private STP is formally defined as follows.

Definition 1: [11] The agent i^{th}'s private STP, S_P^i, as the tuple $\langle V_A^i, E_{NP}^i \rangle$, where both V_A^i and E_{NP}^i are defined above.

Definition 2: [11] The multi-agent shared STP,S_S, is the tuple $\langle V_S, E_S \rangle$, where the set of shared timepoints, $V_S = \bigcup_i V_{AS}^i$ (notice that $v \in V_X^j$ will be included in V_{AS}^i for some i), and where the set of shared edges, $E_S = \{\bigcup_i E_{NS}^i\} \cup \{E_X^i\}$.

Given definitions 1 and 2, now we can redefine Ma-STP as follow:

Definition 3: [11] a Ma-STP, $S = \langle V, E \rangle$, is an union over agents' private STPs, S_P^i, and multi-agent shared STP S_S.

J. Boerkoel and H. Dufree have proposed a full-distributed algorithm for Ma-STP, which has solved the problem by tightening the private STPs at first and then tighten each time-point of Shared-STP without gathering all inter-connected time-points into only one processor, and the privacy of agents' schedule are increased. They have also proposed the distributed algorithm for DMa-STP called DMaTDP that finally generate decoupled Ma-STP [4], [11].

B. Rigidity and Flexibility

In order to evaluate the change of MaSTP by adding decoupling constraints, the useful way is applying an quality matrix, called rigidity that defined as below.

Definition 4: Flexibility [12] given time-points t_i and t_j in a consistent STN, the relative flexibility t_i and t_j is the (non-negative) quantity: $Flex(t_i, t_j) = D(t_i, t_j) + D(t_j, t_i)$.

Rigid Components [12]: Adding a constraint $t_j - t_i \leq \delta$ in the extreme case where $\delta = -D(t_j, t_i)$, causes the updated distance matrix entries to satisfy: $-D(t_j, t_i) = t_j - t_i = D(t_i, t_j)$. In such a case, the temporal difference $t_j - t_i$ is fixed (equivalently, Flex$(t_i, t_j) = 0$), and t_i and t_j are said to be rigidly connected.

Definition 5: Relative Rigidity [12] the relative rigidity of the pair of time-points t_i and t_j in a consistent STN is the quantity: $Rig(t_i, t_j) = \frac{1}{1+Flex(t_i,t_j)} = \frac{1}{1+D(t_i,t_j)+D(t_j,t_i)}$

The rigidity of a consistent STN, S, is the quantity: $Rig(S) = \sqrt{\frac{2}{N(N+1)} \sum_{i<j} [Rig(t_i, t_j)]^2}$.

Because of flexibility is always positive in consistent STP, Flex$(t_i, t_j) \geq 0$, both rigidity component, $Rig(t_i, t_j)$ and Rigidity, $Rig(S)$ are positive and restrict in interval $[0,1]$.

When flexibility of a constraint equal to 0, Flex$(t_i, t_j) = 0$, this means that constraint is complete rigid and its rigid component Rig$(t_i, t_j) = 1$. Similarly, if an STP, S, is completely rigid then Rig$(S) = 1$. At the opposite extreme, if S has absolutely no constraints then $Rig(S) = 0$.

III. Distribute Algorithm for Incrementally Solving DMaSTP

We have proposed a new distributed approach to

incrementally solve a Ma-STP called DI-DMaSTP and has two stages: the first stage tighten and decouple a Ma-STP or return inconsistence, the second stage copes with adding constraints into a tight decoupled Ma-STP that might return inconsistence or new tight decoupled Ma-STP.

In order to tighten a Ma-STP at first stage, we have recommended using the J. Boerkoel's distributed algorithm for Temporal Decoupled problem — DMaTDP [11]. However, the other using scenario is considering the input MaSTP as an empty MaSTP and a set of adding constraints, we have preferred to this scenario.

In the second stage, the algorithm incrementally solves a tight decoupled Ma-STP and it has two part: the first part tighten each agent's private STP by its local adding constraints or return inconsistence, the second part copes with inter-agent adding constraints, in this part, an inter-agent adding constraint is decoupled into two decoupling local constraints which are treated as a local adding constraint above. Finally, it returns inconsistence or new tight decoupled Ma-STP. The process has several main steps as follow:

1) The set of entire adding constraints C is split into each agent adding constraints by the rules that is:

$$C^i = \langle c_{xy} \rangle \mid (x \in V^i)\ or\ (x = z\ and\ y \in V^i)$$

where C^i is the set of adding constraints and V^i is the set of time-points of an agent i^{th}. It is simple to conclude that all C^i have no overlap constraints and union of $C^i\ \forall i$ is C.

2) Each agent has its own processor that checks whether a adding constraint threatened consistence of its private STP and retightens private STP if not.

The problem is solved by separating C^i into different types of adding constraints: adding local constraints C^i_{local} and adding inter-agent constraints C^i_{inter}. An adding local constraint is dealt by running incremental algorithm IFPC on an agent itself, and an adding inter-agent constraint between two agents will be decouple by two decoupling local constraints belong to both agents relates, and then two decoupling local constraints will be deal with by two agents simultaneously.

An inter-agent constraint will be replaced by two local constraints that all involved the reference time-point. We have proposed a decoupling strategy that maximizes the flexibilities of the agents' private STPs. Therefore, a new bound of decoupling constraint is assigned by the middle of possible bounds.

Algorithm 1: Distributed Incremental Decoupled Multigagents STP (DI-DMaSTP)

Inputs: the Agent *i*'s pivate STP instance $S^i = \langle V^i, E^i \rangle$ of an Decoupled MaSTP, a set of adding constraints belong to this agent $C^i = \langle c_{xy} \rangle \mid x \in V^i$.

Outputs: The agent *i*'s private tightening STP of an Decoupled MaSTP or INCONSISTENT.

1: $D^i \leftarrow FPC(S^i)$; // calculate distance graph of private STP

2: if D^i has a negative value then returns inconsistence and halt;

3: $C^i_{local};\ C^i_{inter} \leftarrow$ separate (C^i);

4: for each $c_{xy} \in C^i_{local}$

5: {

6: If c_{xy} has a time point that is not exist in V^i

7: **Extend** (V^i, D^i, c_{xy})

8: return **IFPC** (D^i, c_{xy});

9: }

10: for each $c_{xy} = [-d_{yx}, d_{xy}] \in C^i_{inter}$

11: $d_{yz}; d_{zy} \leftarrow getreferentdistance(y, j)$;

12: If ($w'_{xy} + d_{yz} + d_{zx} < 0$ || $w'_{yx} + d_{xz} + d_{zy} < 0$)

13: return inconsistence and halt;

14: Else

15: {

16: tighten_triangle (*z*, *x*, *y*);

17: $d'_{xz}, d'_{zy}, \leftarrow$

18: $decoupling(d_{xz}, d_{zx}, d_{zy}, d_{yz}, w'_{xy}, w'_{yx})$;

19: tighten_triangle (*z*, *x*, *y*);

20: $d'_{yz}, d'_{zx}, \leftarrow$

21: $decoupling(d'_{xz}, d_{zx}, d'_{zy}, d_{yz}, w'_{xy}, w'_{yx})$;

22: Sent new referent distance (d'_{zy}, d'_{yz}, j);

23: **IFPC** (D^i, d'_{xz}); **IFPC** (D^i, d'_{zx})

24: }

25: $C'_i \leftarrow get\ new\ updated\ constraints$

26: or each $c'_{zx} \in C'_i$

27:return IFPC(S^i, d'_{zx});

Algorithm 2: Extend (V^i, D^i, c_{xy})

Inputs: Agent *i*'s time points, its private distance graph.

Output: Updating agent *i*'s time points, its private distance graph

1: $if\ (x \in V^i\ and\ y \notin V^i)$

2: {

3: for each $v_i \in V_i$

4: {

5: $d_{v_iy} = d_{v_id} + d_{xy}$;

6: $d_{yv_i} = d_{yx} + d_{xv_i}$;

7: add d_{v_iy}, d_{yv_i} into D^i;

8: }

9: add y into V^i;

10: }

11:else

12:{

13: for each $v_i \in V_i$

14: {

15: $d_{v_ix} = \infty$; $d_{v_iy} = \infty$;

16: $d_{xv_i} = \infty$; $d_{yv_i} = \infty$;

17: add d_{v_ix}, d_{xv_i} , d_{v_iy}, d_{yv_i} into D^i;

18: }

19: }

20: add d_{xy}, d_{yx} into D^i;

21: add x, y into V^i;

Algorithm 3: IFPC(D^i, c_{xy});

Inputs: the Agent *i*'s private STP and a local adding constraint.

Outputs: The agent *i*'s new private tightening STP or INCONSISTENT.

1: if ($d_{yx} + w_{xy} < 0$) return INCONSISTENT;

2: else if ($d_{xy} < w_{xy}$) halt;
3: else {
4: $d_{xy} \leftarrow w_{xy}$
5: $I \leftarrow \emptyset; J \leftarrow \emptyset$
6: For all $v \in V_i, v \neq x, v \neq b$ do {
7: If $d_{vy} > d_{vx} + d_{xy}$ **{**
8: $d_{vy} \leftarrow d_{vx} + d_{xy}$
9: $I \leftarrow I \cup v$
10: }
11: If $d_{xv} > d_{xy} + d_{yv}$ **{**
12: $d_{xv} \leftarrow w_{xy} + d_{yv}$
13: $J \leftarrow J \cup v$
14: }
15: }
16: for all $i \in I, j \in J, i \neq j$ do {
17: if $d_{ij} > w_{ja} + w_{aj}$ then
18: $w_{ij} \leftarrow w_{ja} + w_{aj}$
19: **}**
20: Return S

Algorithm 4: decouple $(d_{xz}, d_{zx}, d_{zy}, d_{yz}, w'_{xy}, w'_{yx})$
Inputs: weight of an inter-agent constraint w'_{xy}, w'_{yx} and shortest distances $d_{xz}, d_{zx}, d_{zy}, d_{yz}$.
Outputs: distances four decoupling constraints $d'_{xz}; d'_{zy}$;
1: $d'_{xz} = \frac{1}{2}(d_{xz} - d_{zy}) + \frac{1}{2}w'_{xy}$;
2: $d'_{zy} = \frac{1}{2}(d_{zy} - d_{xz}) + \frac{1}{2}w'_{xy}$;
3: return $d'_{xz}; d'_{zy}$;

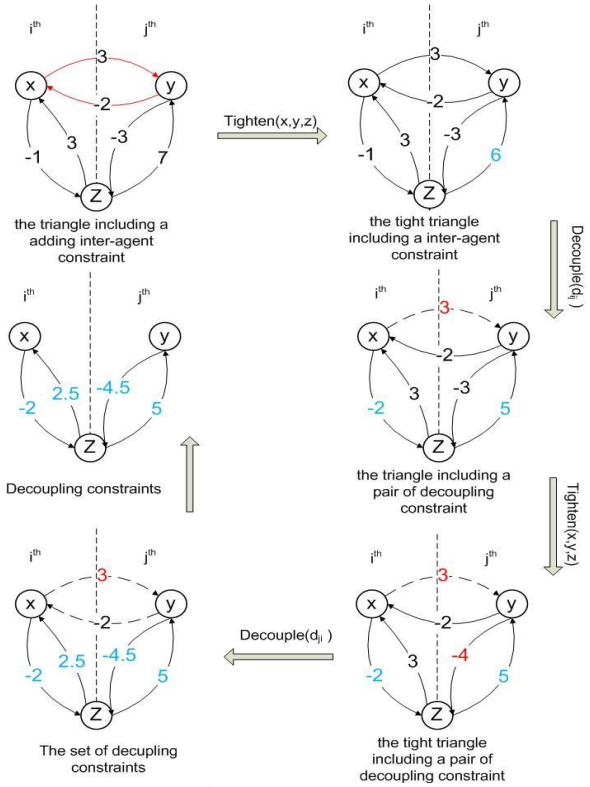

Fig. 1. An example of decoupling a pair inter-agent constraints.

In our point of view, each agent has the right to schedule itself, therefore in the decoupling procedure we have applied the rules to maximize the flexibilities of agents' private STPs. Besides, the roles of both agents are the same then we have balanced the flexibilities of both agents relate to a decoupled intra-agent constraint.

For example, the tighten contraints between two timepoints x, y belong to different agent i^{th} and j^{th} are decoupled by a process depicted in Fig. 1. The value of decoupling constraints are chosen due to algorithm 4 that makes the flexibilities of private decoupling constraints between timepoints $\{z, x\}$ and $\{z, y\}$ are equal: $d'_{zx} + d'_{xz} = d'_{zy} + d'_{yz}$.

IV. The Correctness and the Runtime Shortage

The algorithm DI-DMaSTP deals with adding constraints by separating them into local adding constraints and inter-agent adding constraints, and solves local adding constraints by running Increment Full Path Consistent algorithm — IFPC — on each private STPs and its local adding constraints. And then, an inter-agent adding constraint is decoupled into two local constraints which have bounds set by a decoupling procedure and then two local constraints are dealt by running IFPC in two corresponding private STPs.

The correctness of DIDMaSTP is proved by proving its two minor problems. The first problem is running IFPC for a local adding constraint and a private STP of Decoupled MaSTP is equal to running IFPC for such adding constraint and entire Decoupled Ma-STP. The second one is that an adding inter-agent constraint might threaten the consistence of MaSTP or be decoupled correctly by two local constraints selected by the decoupling procedure.

The algorithm IFPC calculates the shortest path between all pairs of time-points in a STP, and when a constraint is added the algorithm updates the shortest paths of pairs needed. In order to the first problem above, we have to prove that the distance between any pair of time-points belonged to an agent would be evaluated by such agent private STP itself and retightening a local constraint would not effects other agents' private STPs.

Lema 1: The distance of shortest part between two timepoints belong to an agent d_{xy} that is constructed by calculation on only local STPs is also shortest path of such two time-points in entire Decoupled Ma-STP.

Proof: Assume d_{xy} is the distance of the shortest path between x, y in local STPs of agent i^{th}, we have an inequal:

$$d_{xy} \leq d_{xz} + d_{zy} \quad (1)$$

The equal happens if z is a time-point of shortest path.

Assume that there is a shorter path from x to y that involves a timepoint t of other agent j^{th} and has a distance d'_{xy}, then we have an inequality:

$$d'_{xy} < d_{xy} \quad (2)$$

and

$$d'_{xy} = d_{xt} + d_{ty} \quad (3)$$

In Ma-STP, all agents only shared referent time-point z so that the shorter path from x to y through t must involve referent time-point z. Then we have:

$$d'_{xy} = d_{xz} + d_{zt} + d_{tz} + d_{zy} \quad (4)$$

In consistent STP, we have $d_{zt} + d_{tz} \geq 0$, so that

$$d'_{xy} = d_{xz} + d_{zt} + d_{tz} + d_{zy} \geq d_{xz} + d_{zy} \quad (5)$$

From equations [1] and [4] we have $d'_{xy} \geq d_{xy}$ that contradicts to [2] -the assume that d'_{xy} is shorter path from x to y.

The next issue is that running IFPC for a local adding constraint and private STP of Decoupled MaSTP is equal to running IFPC for such adding constraint and entire Ma-STP

Lema 2: In Decoupled MaSTP, the shortest path between two time-points Dij belongs to two different agents is total path of one time point to the referent time-point Diz and the path of the referent time-point to the end Dzj.

Proof: This is consequence of Decoupled MaSTP structure, in which any two private STP only shared referent time-point, therefore all paths between two time-points belong to two different agents have to involve referent time-point, z, including shortest path. Consequently, $D_{ij} = d_{iz} + d_{zj}$.

The soundness of the algorithm will be proved by at first, give a proof that if any local constraint does not threatens it's private STP then it does not threaten the consistent of Ma-STP, and any inter-agent constraint does also.

The second thing we have to prove is that running IFPC in all private STPs will retighten DMa-STP. This is an arbitrary consequence of theorem 1 in J. Boerkoel publication [11], that proves the change of bound of a local constraints will not change other agents' STP.

V. Computing Time Shortage

Besides the purpose of maintaining privacies of agents' scheduling problem on Multi-agent system, our distributed algorithm has improved the computing times also. The time of a distributed algorithm equals to total of calculated time and communicate time (involving time required by the synchronization routine).

In general, computing times of DI-DMaSTP is the maximum computing time of an agent:

$$T^{M}_{DIDMaSTP} = Max\left(T^{M_i}_{DIDMaSTP_i} + T_{C_i}\right)$$

While:

$T^{M_i}_{DIDMaSTP_i}$ is the computing time of the agent i^{th} and is calculated by following equation:

$$T^{M_i}_{DIDMaSTP_i} = 2\left|C^i_{\text{local}}\right|\left(T_{IFPC}(|V_i|) + T_{\text{extend}}(|V_i|)\right) + 2\left|C^i_{\text{inter}}\right|\left(T_{\text{decouple}}(c) + T_{IFPC}(|V_i|)\right)$$

The function "extend" uses a loop to update all shortest paths of exist time-points to a new time point and has the time complexity bound is $\mathcal{O}_{\text{extend}}(|V_i|)$. The function "decouple" just apply several steps to choose a new bound for two decoupling constraints, thus it has the time complexity bound $\mathcal{O}_{\text{decouple}}(\partial)$, where ∂ is a constant parameter. The time complexity of IFPC algorithm is $\mathcal{O}_{IFPC}(n^2)$, with n is the number of time points. Therefore the bound of time complexity of agent i^{th} DI-DMaSTP algorithm is:

$$\mathcal{O}^{M_i}_{DIDMaSTP_i} = 2\left|C^i_{\text{local}}\right|\left(\mathcal{O}_{IFPC}(|V_i|^2) + \mathcal{O}_{\text{extend}}(|V_i|)\right) + 2\left|C^i_{\text{inter}}\right|\left(\mathcal{O}_{\text{decouple}}(c) + \mathcal{O}_{IFPC}(|V_i|^2)\right) = 2\frac{|C_i|}{M}\left(\mathcal{O}_{IFPC}(|V_i|^2)\right)$$

TC_i is the time spent for communication of the agent i^{th} depends on number of its inter-agent constraint C^i_{inter} and the time required to collect information relates to that constraints, called.

$$T_{C_i} = \left|C^i_{\text{inter}}\right|\delta$$

δ is formed by infrastructure of distributed system that run DIFPC algorithm.

Take all into account we have the time complexity of DIFPC in the best case is:

$$\mathcal{O}^{M}_{DIDMaSTP} = max\left(2\frac{|C_i|}{M}\mathcal{O}_{IFPC}(|V_i|^2) + \left|C^i_{\text{inter}}\right|\delta\right)$$

T_{IFPC} is the time complexity of IFPC algorithm and according to L. Plaken (L. Planken, 2008) this algorithm has time complexity upper bound is $\mathcal{O}_{IFPC}(n^2)$-n is total number of time points.

Therefore, in the best case the number of time-point and adding constraints of agents are the same, the time complexity upper bound of the algorithm DIFPC is:

$$\mathcal{O}_{DIFPC} = \mathcal{O}_{FPC}\left(\left(\frac{n}{M}\right)^3\right) + 2\frac{|C|}{M}\mathcal{O}\left(\left(\frac{n}{M}\right)^2\right) + \left|C^i_{\text{inter}}\right|\delta$$

where n is total time-point of MaSTP, M is the number of agents and $|C|$ is total number of adding constraint.

The boundary of runtime when applying IFPC on MaSTP as centralized fashion is $\mathcal{O}_{IFPC} = \mathcal{O}_{FPC}(n^3) + 2|C|\mathcal{O}_{DIFPC}(n^2)$, where $\mathcal{O}_{FPC}$ is the boundary of FPC algorithm that calculates distance matrix and $2|C|\mathcal{O}_{DIFPC}(n^2)$ is the boundary of IFPC algorithm for $2|C|$ adding constraints.

If $\delta$is not so large, then the time complexity of DIFPC algorithm, $\mathcal{O}_{DIFPC}$, is much smaller than time complexity of algorithm IFPC, $\mathcal{O}_{IFPC}$,runs on the same MaSTP as one.

A. Memory Shortage

Our algorithm has outperformance compare to apply Full path consistency algorithm for Ma-STP as one STP thanks to the runtime shortage and memory shortage. The time complexity of DI-DMaSTP is normally shorter than IFPC applying on the same Ma-STP. The number of distance DI-DMaSTP algorithm has to store is also m^2 times smaller than number of distance in the situation using IFPC.

IFPC has to store all pair shortest paths between time-points, then the total number of distances in storage is N^2. Meanwhile, each agent in DI-DMaSTP has to store all pair shortest paths between agents' timepoint, therefore, in average, each agent stores totally $\left(\frac{N}{M}\right)^2$. The total number of distances stored by all agents is only $\frac{N^2}{M}$.

B. Privacy VS Rigidity

Privacy is a major advantage of the DI-DMaSTP algorithm. As we have presented, a MaSTP is partitioned into agents' private and shared components. Privacy problem concern the ability of an agent to restrict its private components and only share external elements within the cooperating sub-group. In DI-DMaSTP algorithm, local constraints of an agent, which involve both private and shared time-points, are eliminated by the agent itself. If any inter-agent constraint has been decoupled due to decoupling

procedure, the new decoupling constraints are only sent to exactly the cooperating agents. Hence, the DI-DMaSTP algorithm does not reveal any of its private time points or constraints, it can be guaranteed that any shared time point are at least kept private in the sub-set of related agents. Moreover, the other strong benefit of the DI-DMaSTP algorithm is that it generates the tight decoupled MaSTP, this means each agent planning now is a part of temporal decoupling STPs, therefore, an agent is able to generate their private particular schedule regardless other agents' schedules.

Decoupling procedure gives the agent an opportunity to self-schedule however it also increases the rigidity of MaSTP. An inter-agent constraint is replaced by two decoupling constraints by the decoupling rules. This is the payoff for privacy.

VI. Experience

A. Data Generator

Running the algorithm on inconsistent instances is not a full test of our algorithm, because inconsistency is discovered earlier than completing decomposition. In addition several testing algorithms have the same mechanism of consistency checking. In order to evaluate complete algorithmic effects, we have to construct a data generator that only generates consistent MaSTP instances and a set of adding constraints that do not threaten the consistence of MaSTP instances. This generator works as a scheduling developer that iteratively adds a new constraint into a consistent STP and is able to evaluate whether a new added constraint maintains consistency of STP or not.

A random problem generator, that is parameterized by the tuple $\langle A, T, P, C_{\text{local}}, C_{\text{Inter}}, \theta \rangle$ where A is the number of agents, T is the number of actions per agent, P is the percentage of its time points that an agent keeps private, C_{local} is the number of local constraints per agent, and C_{Inter} is the total number of the inter-agent constraints, and θ is propotion of adding constraint and existing constraint of an agent.

Particularly, each instance of a problem has A agents, and a number of activities T are added for each agent; an activity has one start time-point and one end time-point. One global zero time-point is created and the distance of all time-points to zero time-point is smaller or equal to 60*T.

For each activity, the lower time bound, lb, is chosen uniformly from the interval $[0, 60]$, and the upper time bound is chosen uniformly in the interval $[lb, lb + 60]$.

Within time-points of each agent, C_{local} local constraints are added randomly but avoiding replacing constraints represented agent's activities. A local constraint between two time-points t_i and t_j is made by choosing uniformly from the tightened intervals $[-D_{ji}, D_{ij}]$.

Besides a number internal constraints the number C_{Inter} external constraints are also added over all agents. We first randomly select a number, $P * T * 2$, of shared time points for each agent. Then choose uniformly a shared time-point t_i of agent i^{th} and then choose uniformly an agent j^{th} differ from agent i^{th}, choose uniformly a shared time point t_j of agent j^{th} and make a new constraint between two time-points t_i and t_j by setting its bound uniformly from the tighten intervals $[-D_{ji}, D_{ij}]$. We repeat C_{Inter} times for each instance MaSTP, and accept that a new constraint might replace old one.

The proportion of adding constraint and existing constraint of an agent θ is chosen between [0, 1]. And then, for each agent, the generator has put θC_{local} local adding constraints and θC_{inter} inter-agent adding constraints with the same method above.

Considering the consistent property of a generated graph, we have seen that adding actions and a zero time point will not threaten consistent property of the graph. Moreover, adding extra local constraints and external constraints also satisfy the rule that adding a new constraint between two time-points t_i and t_j, $t_j - t_i \leq \delta$ such that $-D_{ji} \leq \delta$, therefore new adding constraint will not threaten the consistency. Taking above reasons into account; we can conclude that the generated graph is consistent.

B. Experiment

We have firstly tested DI-DMaSTP algorithm with the size of problem. Therefore, the generator makes the set of MaSTP instances with the tuple of parameter, $\langle A, T, P, C_{\text{Intra}}, C_{\text{Inter}}, \theta \rangle$, where the number of agents is increasing, A = {2,4,6,8,10,13,15}. Each agent has T = 10 actions, and we assume that they have least as many joint actions as private actions, we set P = 0.4. Finally, we know that algorithms constructing a decomposition of STP based on PPC only work well with sparse STNs; therefore we set number of added local constraint $C_{\text{Intra}} = 2 * T$; number of inter-agent constraints $C_{\text{Inter}} = 2T * P * A * 3$, and proportion of adding constraints θ = 0.4.

Secondly, we have evaluated our algorithms with respect to the proportion of shared time point in total time point of agent, this also reflects the proportion between adding local constraints and adding inter-agent constraints. We generate the set of MaSTP instance with the tuple of parameter, $\langle A, T, P, C_{\boldsymbol{Intra}}, C_{\text{Inter}} \rangle$, where A = 10, T = 10. Proportion, P, is assigned a value from the set {0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8}. The number of intra-agent constraints, C_{Intra}, is double the time points; the inter-agent constraints, C_{Inter}, and proportion of adding constraints θ are the same as in previous section

We have run all algorithm in multi-core configuration on machine have Intel® Core™ i5 2.5GHz with 4Gb of memory.

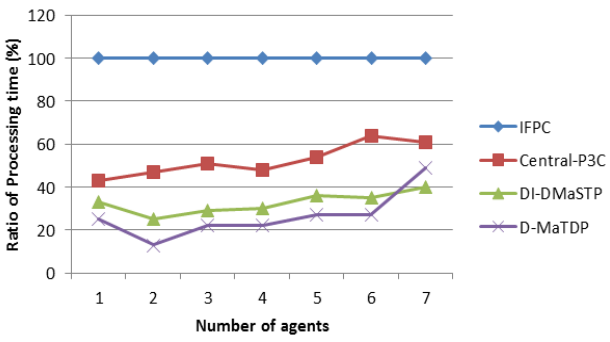

Fig. 2. Ratio of processing time vs number of agent.

In processing time aspect, the experiment data points out that DI-DMaSTP algorithm is much faster than IFPC algorithm, the former has the average processing time is 32.6

percent of later' one. The DI-DMaSTP algorithm is nearly the same distributed algorithm D-MaTDP, however, D-MaTDP does not has ability to deal with adding constraints.

The observed data also give an interesting property, in the first stages, when the number of agents increases the ratio of processing time of distributed algorithms decreases, however when number of agent, A, reaches to a specific value this ratio decreases (Fig. 2). When the number of agents increases, the communication is more complex and requires more time to synchronize between agents and communication shorts the benefit of computing time.

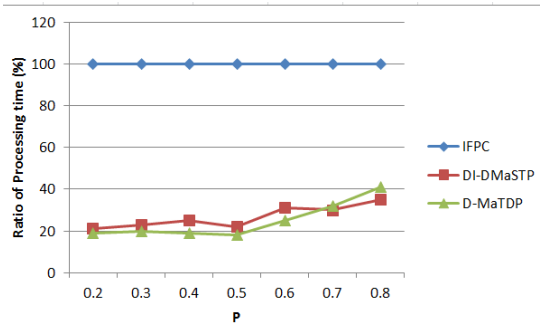

Fig. 3. Ratio of processing time vs proportion of shared time points *P*.

The algorithm DI-DMaSTP and D-MaTDP only process the adding local constraints completely synchronously, so their performances are strongly depended on the proportion between local and inter-agent constraints, and in our test this proportion dominated by the proportion of private and shared time points, P. When proportion, P, increases the processing time of DI-DMaSTP algorithm increases and comes closer to the processing time of IFPC algorithm (Fig. 3).

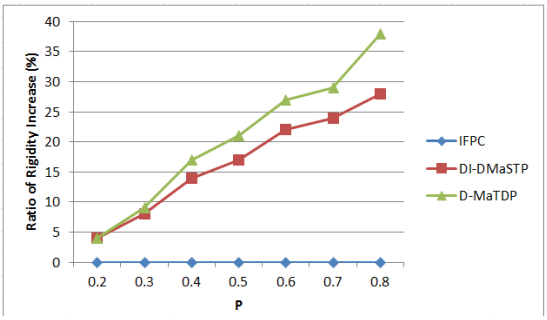

Fig. 4. Ratio of rigidities of algorithms and input vs proportion of shared time point.

Algorithm IFPC generates a decomposition of the MaSTP, therefore its rigidity are equal to rigidity of the input MaSTP instance, obviously. Rigidity of DI-DMaSTP and D-MaTDP algorithms are higher than the input MaSTP and depend on number of inter-agent constraints. Particularly, when the proportion of shared time points, P, increases from 0.2 to 0.8 the proportion of rigidity of DIFPC and input MaSTP increases from 4% to 38%, while DI-DMaSTP increases from nearly 4% to 28% (Fig. 4).

VII. Conclusions

Applying the distributed approach and the Full path consistency algorithm in DI-DMaSTP to incrementally solving MaSTP avoid the limitation of algorithms based on chordal graph, for example IP3C [13], that only works with an adding constraint does not threaten chordal property. Therefore, all adding constraints are solved in. The proposed algorithm also maintains the privacy as planning for all agents. Each agent would attend in many different organizations in reality, therefore maintaining privacy of agent schedule is a strong advantage.

The DI-DMaSTP algorithm also has outperformance compare to apply Full path consistency algorithm for Ma-STP as one STP thanks to the runtime shortage and memory shortage. The time complexity of DI-DMaSTP is normally shorter than IFPC applying on the same Ma-STP. The number of distance DI-DMaSTP has to stored is also m^2 times smaller than number of distance in the situation of using IFPC.

The disadvantage of DI-DMaSTP is that it increases the rigidity of entire MaSTP,that means it miss a part of available solution schedules. In our algorithm, the decoupling process manages to maximize the flexibilities of private STP then it limits the increasing of rigidity of private STP as much as possible. In the future, we are going to to apply new heuristic algorithm to choose better bounds of decoupling constraints, which decrease rigidity of entire MaSTP.

References

[1] J. Boerkoel and H. Dunfee, "Distributed algorithm for solving the multiagent temporal decoupling problem," *Artificial Intelligence*, pp. 141-148, 2011.
[2] K. S. Koubarakis, "Backtracking algorithms," *Artificial Intelligence*, vol. 120, pp. 81-117, 2000.
[3] L. R. Planken, "Incrementally solving the STP by enforcing partial path consistency," *PlanSIG-08*, pp. 87-94, 2008.
[4] R. Dechter, *Constraint Processing*, San Francisco, CA, USA: Morgan Kaufmann Publisher Inc, 2003.
[5] R. Dechter and I. Meiri, "Temporal constraint network," *Arificial Intellegence*, pp. 61-95, 1991.
[6] L. Xu and B. Choueiry, "A new efficient algorithm for solving the simple temporal problem," *TIME-ICTL-03*, pp. 210-220, 2003.
[7] C. Bilieck *et al*., "Path Consistency on triangulated constraint graphs," *IJCAI*, pp. 456-461, 1999.
[8] J. R. S. Blair *et al*., *An Introduction to Chordal Graphs and Clique Trees*, New York: Springer-Verlag, 1993.
[9] U. Kjaerulff, "Graph triangulation — algorithms giving small total state space," Technical Report, Aalborg: University of Aalborg, 1990.
[10] L. R. Planken *et al*., "P3C-New algorithms for the simple temporal problem," *ICAPS*, pp. 256-263, 2008.
[11] J. Boerkoel and H. Dunfee, "A comparison of algorithms for solving the multiagent simple temporal problem," *ICAPS*, pp. 27-33, 2010.
[12] L. Hunsberger, "Algorithms for a temporal decoupling problem in multi-agent planning,"*AAAI*, pp. 468-475, 2002.
[1] L. R. Planken and M. M. de Weerdt, "Optimal temporal decoupling in multiagent systems," *AAMAS*, pp. 789-796, 2010.

Cu Nguyen Giap was born in 1984 in Phutho province, Vietnam. He received the B.Sc. degree in information technology from Hanoi University of Technology in 2007. In 2012, he received the M.Sc. degree in computer science from Vrije Universiteit Brussels. Now, He is a lecturer in the Faculty of Economic Information System in Vietnam Commercial University. His research interests include scheduling algorithm, expert & prediction system, parallel & genetic algorithm, neutral network.

Hien Thu Thi Do was born in 1980 in Bacgiang province, Vietnam. In 2008, She received the M.S. degree in information management from Shute University, Taiwan. Now, She is a lecturer in the Faculty of Economic Information System in Vietnam Commercial University. Her research interests include scheduling algorithm, information management system, expert & prediction system.