Nonserial Dynamic Programming with Applications in Smart Home Appliances Scheduling – Part I: Precedence Graph Simplification

Kin Cheong Sou, Henrik Sandberg and Karl Henrik Johansson

***Abstract*— In this and a companion paper a dynamic programming (DP) approach to solve a smart home appliances scheduling problem is considered. The challenge with solving the scheduling problem is the coupling of decision variables due to some time precedence constraints. In general, the system of precedence constraints may contain redundant constraints that offer opportunities for simplification. This simplification is desirable for reducing the computation effort of the nonserial DP procedure presented in the companion paper (i.e., Part II). The current paper establishes the uniqueness of the maximum set of redundant constraints and its polynomial-time solvability with optimality guarantee, under the sufficient and necessary condition that the precedence graph (a graph representation of the precedence constraints system) does not contain any cycle with nonnegative weight. A numerical case study indicates the efficiency of the proposed simplification algorithm versus the brute-force enumerative search. Besides helping to reduce the computation effort in the DP procedure described in the companion paper, the algorithm in the current paper solves a generalization of a precedence graph simplification problem arising from application areas such as parallel computing.**

I. Introduction

A major goal of smart grid technology (e.g., smart meters) is to provide consumers with demand response signals such as electricity tariff and CO_2 footprint so that the consumers can consciously control their energy consumption patterns. These demand response signals provide incentives for the consumers to help reduce peak energy demand by load balancing, as this is particularly relevant in a situation with high level of renewable energy penetration. However, the volume of information of the time-varying tariff and CO_2 demand response signals can be overwhelming for the consumers. Therefore, optimal appliances scheduling is a key component for the automatic decision supporting technologies to enable smart grid realizability. Recently, the appliances scheduling problem to exploit demand response signals has received much attention (e.g., [1]–[7]).

This paper describes the extension of the dynamic programming (DP) approach to the smart home appliances scheduling problem considered in [1], [2]. The problem seeks to determine the start times, denoted by t_i for $i = 1,2,\ldots,n$, of the energy phases of the appliances. The start times t_i are discrete (i.e., $t_i \in \{0,\Delta t, 2\Delta t, \ldots (N-1)\Delta t\}$ for some $\Delta t > 0$ and positive integer N). In addition, it is assumed that $L_i \le t_i \le U_i$ for all i, and L_i and U_i are given ($0 \le L_i \le U_i$). It is possible to have $L_i = -\infty$ and/or $U_i = \infty$ to indicate that the corresponding constraints are in fact not present. Further, the start times must satisfy time precedence constraints of the form $l_{ij} \le t_j - t_i \le u_{ij}$ for all $\{i,j\}$ pairs, where l_{ij} and u_{ij} are given numbers with $l_{ij} \le u_{ij}$. Again, it is allowed to have $l_{ij} = -\infty$ and/or $u_{ij} = \infty$ to indicate that the corresponding constraints are in fact not present. Without loss of generality, it is assumed that all L_i, U_i, l_{ij} and u_{ij} are integer multiples of Δt. The time precedence constraints can model the time precedence relationships between energy phases within and between appliances. They can also handle minimum and maximum delays between energy phases. In addition, the schedule is subjected to a resource budget constraint: $\sum_{i=1}^{n} g_i(t_i) \le b$ for some arbitrary given functions g_i and given budget b. It is assumed that the values of g_i and b are quantized. That is, $g_i(t_i)$ and b are nonnegative integer multiples of some given $\Delta b > 0$. Examples of the resource in the budget constraint include financial cost or CO_2 cost. The budget constraint is imposed, for instance, when the ε-constraint method is used to compute the Pareto optimal solution in the electricity bill/CO_2 cost trade-off problem [2]. The objective of the scheduling problem is to minimize the sum of $f_i(t_i)$ (e.g., electricity cost or CO_2 cost). The objective function components f_i are given but arbitrary. The optimization problem can be summarized as follows:

$$\begin{aligned} \underset{t_1,t_2,\ldots,t_n}{\text{minimize}} \quad & \sum_{i=1}^{n} f_i(t_i) \\ \text{subject to} \quad & l_{ij} \le t_j - t_i \le u_{ij}, \quad \forall\, \{i,j\} \\ & L_i \le t_i \le U_i, \quad \forall\, i \\ & \sum_{i=1}^{n} g_i(t_i) \le b. \end{aligned} \tag{1}$$

In [2] a DP approach was reported to solve (1) when the time precedence constraints are serial. That is, there exists a permutation $i_1, i_2, \ldots, i_n$ of $1,2,\ldots,n$ such that the time precedence constraints can be written as the following at most $n-1$ constraints:

$$\begin{aligned} l_{i_1 i_2} \quad &\le \quad t_{i_2} - t_{i_1} \quad \le \quad u_{i_1 i_2}, \\ l_{i_2 i_3} \quad &\le \quad t_{i_3} - t_{i_2} \quad \le \quad u_{i_2 i_3}, \\ &\vdots \\ l_{i_{n-1} i_n} \quad &\le \quad t_{i_n} - t_{i_{n-1}} \quad \le \quad u_{i_{n-1} i_n}. \end{aligned} \tag{2}$$

Again, the u's and l's in (2) can be set to $+\infty$ and/or $-\infty$ respectively, depending on whether the specific constraints exist or not. Any set of time precedence constraints not in the form of (2) is referred to as nonserial.

Kin Cheong Sou is with the Department of Mathematical Sciences, Chalmers University of Technology and the University of Gothenburg, Sweden. The other authors are with the ACCESS Linnaeus Center and the Automatic Control Lab, the School of Electrical Engineering, KTH Royal Institute of Technology, Sweden. kincheong.sou@chalmers.se, {hsan,kallej}@kth.se

While standard DP can handle problem (1) with serial time precedence constraints, it is not obvious whether it is possible or in what manner the standard DP can be applied to the general problem with nonserial time precedence constraints [8]. The main objective of this and the companion paper [9] is to describe a modified DP approach to solve (1) with nonserial time precedence constraints. The approach should provide the exact optimal solution to (1), and the computation should be as efficient as possible.

A. Problem Statement

In this paper a procedure to simplify the system of time precedence constraints in (1) is described. The procedure identifies redundant time precedence constraints that are implied by other time precedence constraints in the system. For example, in the following system of constraints the first one is implied by the later two: $t_3 - t_1 \geq 3$, $t_3 - t_2 \geq 2$, $t_2 - t_1 \geq 2$. In general, given a system of time precedence constraints

$$l_{ij} \leq t_j - t_i \leq u_{ij}, \quad \forall\{i,j\}, \tag{3}$$

where l_{ij} and u_{ij} are given multiples of Δt. The simplification problem seeks a more compact equivalent system of time precedence constraints

$$\tilde{l}_{ij} \leq t_j - t_i \leq \tilde{u}_{ij}, \quad \forall\{i,j\}, \tag{4}$$

by finding appropriate values of $\tilde{l}_{ij}$ and $\tilde{u}_{ij}$, so that under the constraints that

1) $\tilde{l}_{ij} = l_{ij}$ or $\tilde{l}_{ij} = -\infty$, for all $\{i,j\}$,
2) $\tilde{u}_{ij} = u_{ij}$ or $\tilde{u}_{ij} = \infty$, for all $\{i,j\}$,
3) (3) and (4) define the same constraint set,

the number of $\tilde{l}_{ij}$ and $\tilde{u}_{ij}$ setting to their respective extreme values (i.e., $-\infty$ or $+\infty$) is maximized. In order words, (3) and (4) define the same constraint set, while in (4) the maximum number redundant constraints are removed.

One motivation for the simplification is that by describing (1) with a more compact system of time precedence constraints, the computation effort for the nonserial DP procedure presented in the companion paper [9] can be reduced. As an alternative motivation, the time precedence constraint system simplification problem is a generalization of a precedence relation simplification problem considered in the literature (e.g., parallel computing [10]–[12] and manufacturing system [13], [14]).

B. Contribution and Organization of the Paper

In Section II a directed precedence graph representation of the time precedence constraint system in (3) is described. Some properties of the precedence graph are discussed. In particular, it will be argued that the precedence graph naturally does not contain any nonnegative weight cycle, provided that the degeneracy in the precedence graph is resolved. The simplification problem associated with (3) and (4) can be written as a precedence graph simplification problem in which the maximum redundant edge set is sought. Section III describes the main contribution of the paper. It is shown that under the no-nonnegative-weight-cycle condition, the maximum redundant edge set is unique. In addition, this paper establishes that the maximum redundant edge set can be found in polynomial-time by providing a polynomial-time solution algorithm based on Bellman-Ford longest path computation (for graphs without nonnegative weight cycles). Thus, the computationally intensive enumerative search for the maximum redundant edge set can be avoided. In the end of Section III, it is shown that the no-nonnegative-weight-cycle condition is also necessary for the uniqueness of the maximum redundant edge set. This is established by a counterexample. In Section IV a numerical case study demonstrates the effectiveness of the proposed algorithm for precedence graph simplification.

II. Directed Precedence Graph Representation of Precedence Constraints System

To proceed with the simplification, a directed graph representation of the time precedence constraint system in (1) needs to be described first. These constraints can always be written as $t_v - t_u \geq c_{uv}$ with appropriately defined indices $u \in \{1,2,\ldots,n\}$ and $v \in \{1,2,\ldots,n\}$ and weight c_{uv}. For instance, the constraint $l_{ij} \leq t_j - t_i \leq u_{ij}$ can be written as $t_j - t_i \geq l_{ij}$ and $t_i - t_j \geq -u_{ij}$. The time precedence constraint system can be described by a directed time precedence graph $G = (V,E)$ with edge weights. G and the weights are defined by

- The set of all nodes, denoted by V, is defined as $V = \{1,2,\ldots,n\}$. Each node in V corresponds to a start time decision variable in (1).
- The set of all weighted directed edges, denoted by E, is defined as the set of all ordered pairs (u,v) for $u \in V$ and $v \in V$ such that $(u,v) \in E$ with weight c_{uv} if and only if the time precedence constraint $t_v - t_u \geq c_{uv}$ is present (i.e., $c_{uv} > -\infty$).

A time precedence graph G cannot be arbitrary. For instance, G does not have any self-cycles (i.e., edges originating and ending at the same node). Also, it is assumed that G does not have parallel edges with the same direction connecting two nodes, since one edge will imply the rest. Furthermore, G does not have positive weight directed cycles:

Lemma 1: Let the sequence of nodes $i_0, i_1, i_2, \ldots, i_m$ denote a directed cycle in the time precedence graph G, with $(i_k, i_{k+1}) \in E$ for $k = 0,\ldots,m-1$ and $i_k \neq i_j$ for $k \neq j$ except $i_0 = i_m$. Then the weight of the cycle, defined as $c_{i_0 i_1} + c_{i_1 i_2} + \ldots + c_{i_{m-1} i_m}$, must be nonpositive.

Proof: The time precedence constraints corresponding to all edges in the cycle are listed as

$$\begin{array}{rcl} t_{i_1} - t_{i_0} & \geq & c_{i_0 i_1} \\ t_{i_2} - t_{i_1} & \geq & c_{i_1 i_2} \\ & \vdots & \\ t_{i_m} - t_{i_{m-1}} & \geq & c_{i_{m-1} i_m}. \end{array} \tag{5}$$

Summing up all inequalities, with the fact that $i_0 = i_m$, leads to the desired inequality that $0 \geq c_{i_0 i_1} + c_{i_1 i_2} + \ldots + c_{i_{m-1} i_m}$. ■

Remark 1: Lemma 1 does not exclude the cases where cycles in a time precedence graph G have zero weights.

However, this paper makes a further assumption that zero weight cycles are not allowed (i.e., all cycles have negative weights). This assumption amounts to the removal of degeneracy which does not affect the optimal solution of problem (1). The details can be found in Appendix A. The assumption of cycles having negative weights is critical in the systematic approach to simplify G. This will be clear in the subsequent discussions.

Because of the correspondence between the time precedence constraint and the edge of the time precedence graph, in the sequel the simplification of the time precedence constraint system will be discussed in terms of the simplification of the corresponding precedence graph.

III. Simplification of Precedence Graph

To describe the systematic procedure to simplify the time precedence graph G, some definitions are needed. In G, a (directed) *walk* from node u to node v is described by a sequence of traversed nodes $u = i_0, i_1, \ldots, i_m = v$, where $(i_k, i_{k+1}) \in E$ for all k. A *closed (directed) walk* is a walk with $i_0 = i_m$. A (directed) *path* from u to v is a walk from u to v with the additional requirement that all traversed nodes are distinct. A (directed) *cycle* is a closed walk where $i_0, i_1, \ldots, i_{m-1}$ are distinct and $i_0 = i_m$. If w denotes a walk (including path and cycle), then the weight of w, denoted by c_w is $c_w = c_{i_0 i_1} + c_{i_1 i_2} + \ldots + c_{i_{m-1} i_m}$. That is, the weight of the walk is the sum of the weights of the traversed edges, with the edge weight added as many times as an edge is traversed. An edge set $E_r \subset E$ is referred to as *a set of redundant edges* (or *redundant edge set*) if it satisfies the following condition:

$$\text{For all } (u,v) \in E \text{ with weight } c_{uv}, \text{ there exists a path } p_{uv} \text{ in } (V, E \setminus E_r) \text{ from } u \text{ to } v \text{ such that } c_{p_{uv}} \geq c_{uv}. \tag{6}$$

There can be multiple sets of redundant edges in a time precedence graph. However, the redundant edge set with the maximum number of edges, denoted by $E_r^\star$, is unique. The following statement and its corollary provide the rationale:

Lemma 2: Let E_r^1 and E_r^2 both be redundant edge sets satisfying (6). Then $E_r^1 \cup E_r^2$ is also a redundant edge set satisfying (6).

Proof: To verify that $E_r^1 \cup E_r^2$ is indeed a redundant edge set, it is sufficient to verify that for each $(u,v) \in E_r^1 \cup E_r^2$ with weight c_{uv}, it holds that

$$\exists \text{ a path } p_{uv} \text{ in } (V, E \setminus (E_r^1 \cup E_r^2)) \text{ from } u \text{ to } v \text{ s.t. } c_{p_{uv}} \geq c_{uv}. \tag{7}$$

Without loss of generality, assume that $(u,v) \in E_r^1$ (otherwise the labels can be exchanged). Then by (6) there exists a walk (which is in fact a path) w_1 in $(V, E \setminus E_r^1)$ from u to v such that $c_{w_1} \geq c_{uv}$. If w_1 is also in $(V, E \setminus E_r^2)$ then w_1 belongs to $(V, E \setminus (E_r^1 \cup E_r^2))$ and (u,v) satisfies (7). If, on the other hand, w_1 contains edges belonging to E_r^2, then by (6) each of these edges can be replaced by a corresponding path in $(V, E \setminus E_r^2)$. Also by (6), the weights of the replacement paths are no less than the weights of the corresponding edges. This results in another walk w_2 in $(V, E \setminus E_r^2)$ from u to v, with possible edges in E_r^1. The walk w_2 has strictly more nodes (and edges) than w_1, and $c_{w_2} \geq c_{w_1}$. If w_2 is in $(V, E \setminus E_r^1)$ then (u,v) satisfies (7), as argued above. Otherwise, the process of finding replacement walks $w_3, w_4, \ldots$ with increasing number of nodes and nondecreasing weights would continue. This process terminates if and only if a replacement walk from u to v is found, which is entirely in $(V, E \setminus (E_r^1 \cup E_r^2))$. Let $w^\star$ denote such a replacement walk (will be shown to exist). The proof will be completed (for edge (u,v)) if the following two claims can be asserted: (a) the replacement-path-finding process would terminate in a finite number of iteration with some $w^\star$, and (b) upon termination of the replacement-path-finding process the replacement walk $w^\star$ can be used to show that (u,v) satisfies (7). To see claim (a), assume on the contrary that the replacement-walk-finding does not terminate. It is noted that any walk from u to v can be decomposed into a path from u to v and a finite number of cycles (see Appendix B for a proof). In addition, in a finite graph the numbers of possible paths and cycles are finite. Therefore, by the pigeonhole principle, at some iteration in the replacement-walk-finding process will generate a walk $\tilde{w}$ traversing a cycle, and one of the previously constructed walks is exactly the same as $\tilde{w}$ except that the cycle is not traversed. The construction of the walks implies that the cycle has nonnegative weight. This contradicts the assumption that all cycles in G have negative weights. This shows the finite termination of the replacement walks construction process, and this implies the existence of the replacement walk $w^\star$ from u to v, entirely in $(V, E \setminus (E_r^1 \cup E_r^2))$, because the existence is the sufficient and necessary condition for finite termination. Hence, claim (a) is established. Finally, to see claim (b), note again that $w^\star$ can be decomposed into a path from u to v and a finite number of cycles. Removing all cycles in the walk $w^\star$ will result in a path from u to v which is entirely in $(V, E \setminus (E_r^1 \cup E_r^2))$. In addition, the weight of this path is greater than that of $w^\star$, from which the path is constructed because all cycles are assumed to have negative weights. Hence (u,v) satisfies (7), and claim (b) is established. Applying the same proof to all members of $E_r^1 \cup E_r^2$ completes the proof. ■

Lemma 2 implies that the maximum redundant edge set is unique:

Corollary 1: $E_r^\star$, the redundant edge set with the maximum number of members, contains all redundant edge sets as subsets. Consequently, $E_r^\star$ is unique.

Proof: Let $E_r^\star$ denote a maximum redundant edge set. Suppose that there exists a redundant edge set E_r such that $E_r \setminus (E_r \cap E_r^\star) \neq \emptyset$. Then Lemma 2 implies that $E_r^\star \cup E_r$, which has more members than $E_r^\star$, is also a redundant edge set satisfying (6). This contradicts the assumption that $E_r^\star$ is a maximum redundant edge set. Hence, E_r does not exist and the desired statements hold. ■

With the uniqueness asserted by Corollary 1, finding the maximum redundant edge set $E_r^\star$ becomes the remaining goal for simplification. Lemma 2 also leads to a polynomial time algorithm to find $E_r^\star$, avoiding the need for expensive enumerative search over the power set of E.

Algorithm 1 (finding $E_r^\star$)

1) For each $(u,v) \in E$, define $G_{uv} := (V, E \setminus \{(u,v)\})$ with edge weights c_{ij} for all $(i,j) \in E \setminus (u,v)$. Solve a longest path problem on G_{uv} with u and v being the origin and destination, respectively. The longest path problem can be solved using, for example, the Bellman-Ford algorithm (e.g., [15]) in polynomial time because all cycles have negative weights. If the longest path weight is greater than or equal to c_{uv}, then set $E_r^{uv} = \{(u,v)\}$. Otherwise set $E_r^{uv} = \emptyset$.
2) Set $E_r^{\max} := \bigcup_{(u,v)\in E} E_r^{uv}$, and $E_r^{\max}$ is the desired maximum redundant edge set. That is, $E_r^{\max} = E_r^\star$, as it will be shown in Theorem 1.

Remark 2: The computation effort of **Algorithm 1** is $O(|V||E|^2)$ because for each $(u,v) \in E$ the Bellman-Ford algorithm is applied, requiring $O(|V||E|)$ units of basic computation [15].

Theorem 1: Let $E_r^{\max}$ be returned by **Algorithm 1**, and $E_r^\star$ denote the maximum redundant edge set satisfying (6), then $E_r^{\max} = E_r^\star$.

Proof: If $(u,v) \in E_r^\star$, then by (6) there exists a path p_{uv} in $(V, E \setminus E_r^\star)$ connecting u to v with $c_{p_{uv}} \geq c_{uv}$. p_{uv} is also in $(V, E \setminus \{(u,v)\})$ because $\{(u,v)\} \subset E_r^\star$. Hence $(u,v) \in E_r^{uv} \subset E_r^{\max}$ as defined in the algorithm. This shows that $E_r^\star \subset E_r^{\max}$. On the other hand, since E_r^{uv} for all $(u,v) \in E$ are redundant edge sets and by Lemma 2 $E_r^{\max} = \bigcup_{(u,v)\in E} E_r^{uv}$ is also a redundant edge set satisfying (6), $E_r^{\max} \subset E_r^\star$ by Corollary 1. ∎

Remark 3: References [10]–[14] consider the special case of the maximum redundant edge set problem where the edge weights are all zero. Except for [14], the above references do not investigate the simplification problem from the graph point of view. However, [14] considers only directed acyclic graphs with zero edge weights. In addition, there is no proof in the presented algorithm (i.e., Figure 9) in [14].

Remark 4: **Algorithm 1** presented in this paper can identify the maximum redundant edge set for any directed graph with arbitrary edge weights as long as all directed cycles have negative weights. The graph does not need to be a time precedence graph as motivated in this paper. The assumption of negative weight cycles is sufficient and necessary for the presented algorithm to be correct. In particular, it is possible to construct an example with nonnegative weight cycles where the maximum redundant edge set is not unique. See Figure 1 for an illustration.

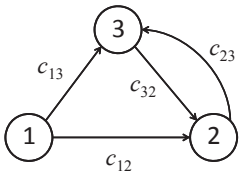

Fig. 1. A 3-node example which is not a time precedence graph. Consider the case where the weights $c_{ij} = 1$ for all edges. There are two redundant edges, namely $(1,2)$ and $(1,3)$. However, $\{(1,2),(1,3)\}$ is not a redundant edge set according to (6). The cycle $(2,3,2)$ has positive weight being two.

IV. Numerical Illustration of Precedence Graph Simplification

To demonstrate the effectiveness of **Algorithm 1**, a numerical case study is carried out. All computation is performed on a PC with 2.5GHz CPU and 8GB of RAM, using MATLAB and MatlabBGL [16] for the Bellman-Ford algorithm. The precedence graph is a 6-node complete directed graph whose topology is shown in Figure 2 and the corresponding edge weights are in the adjacency matrix C defined in (8).

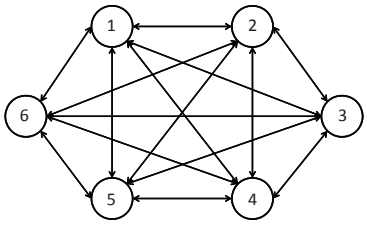

Fig. 2. The topology of the example time precedence graph, which is a six-node complete directed graph.

$$C = \begin{bmatrix} \infty & -15 & -3 & 8 & 7 & 8 \\ 10 & \infty & 12 & 23 & 22 & 23 \\ 1 & -14 & \infty & 11 & 10 & 11 \\ -13 & -29 & -12 & \infty & -1 & 0 \\ -16 & -31 & -12 & -7 & \infty & 1 \\ -12 & -27 & -12 & -5 & -8 & \infty \end{bmatrix}. \tag{8}$$

In (8), the symbol ∞ in the (i,j) entry means edge (i,j) does not exist (e.g., there is no self cycle). It can be verified that all cycles in the example precedence graph have negative weights (see Appendix C for the construction of the example precedence graph). Applying **Algorithm 1** to the example precedence graph results in a simplified precedence graph in Figure 3. It can be verified that the removed edges in

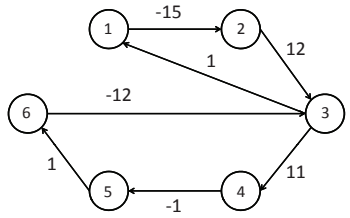

Fig. 3. The resulted simplified precedence graph after applying **Algorithm 1** to the example precedence graph in Figure 2 with edge weight adjacency matrix defined in (8). The numbers on the edges denote the corresponding edge weights.

Figure 3 (from those in Figure 2) form a redundant edge set according to (6). In addition, to verify the optimality of the simplification due to **Algorithm 1**, an enumerative search of the maximum redundant edge set over the power set of the set of all edges is implemented and applied to this example. It is found that the enumerative search results in the same simplified precedence graph as shown in Figure 3.

To compare the computation efficiency, **Algorithm 1** and the enumerative search are applied to random instances of complete precedence graphs and the computation time

is recorded. For each $n = 2,3,4,5$, one hundred random instances of n-node weighted complete graph are generated, where the edge weights are randomly chosen such that all cycles (if any) have negative weights. The details of the construction of the random graph instances can be found in Appendix C. Figure 4 shows the average and maximum computation times due to **Algorithm 1** and the enumerative search. It clearly indicates the efficiency of **Algorithm 1**. Finally, it is verified that in all cases **Algorithm 1** and the enumerative search result in the same maximum redundant edge sets.

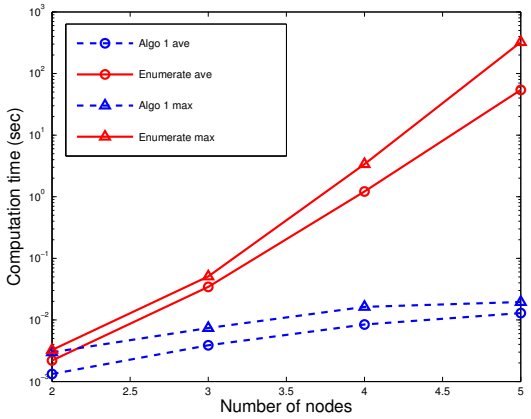

Fig. 4. Computation times for solving random instances of complete precedence graph with n nodes. This figure clearly indicates the computation efficiency of **Algorithm 1** over the enumerative search.

V. Conclusions

While it has been known that the maximum redundant edge set problem can be solved in polynomial-time when the edge weights of the precedence graph are all equal to zero, this paper extends the polynomial-time complexity result to the case with arbitrary edge weights as long as all cycles have negative weight. This paper also shows that precedence graphs naturally do not have any nonnegative weight cycles, as long as the degeneracy associated with the zero weight cycles is resolved. Under this no-nonnegative-weight-cycle condition, it can be established that the maximum redundant edge set is unique, and its computation can be based on polynomial-time Bellman-Ford longest path algorithm for graphs without nonnegative weight cycles. The no-nonnegative-weight-cycle condition is analogous to the no-negative-length-cycle condition for the polynomial complexity of the shortest path problem.

Appendix A

A zero weight cycle implies degeneracy in which only one (arbitrary) start time among the start times involved in the cycle is independent. To see this, summing up all constraints except the first one in (5) results in $t_{i_0} - t_{i_1} \geq -c_{i_0 i_1}$. This inequality, together with the first constraint $t_{i_1} - t_{i_0} \geq c_{i_0 i_1}$, implies that $t_{i_1} - t_{i_0} = c_{i_0 i_1}$. Similarly, it can be shown that $t_{i_{k+1}} - t_{i_k} = c_{i_k i_{k+1}}$ for all k. Therefore, the zero weight cycle can be collapsed to a single "supernode". For any outside node u, for any edge between u and a member of the supernode, there is an edge connecting u and the supernode with appropriate adjustment of the weight to reflect the corresponding adjustment to the time precedence constraint. For example, suppose i_0 is chosen to be the independent start time (i.e., it represents the supernode) and there exists an edge (u, i_k) with weight $c_{i_k u}$ in G. Then there should be an edge (u, i_0) with weight $c_{i_k u} + \sum_{s=0}^{k-1} c_{i_s i_{s+1}}$.

To identify the zero weight cycles, the Bellman-Ford shortest path algorithm (e.g., [15]) can be applied to a modified graph G'. G' contains the same node and edge sets of G. However, if (u,v) is an edge in G with weight c_{uv}, then in G' the corresponding weight for (u,v) is $-c_{uv} - \varepsilon$, where $0 < \varepsilon < \frac{\Delta t}{n}$. By construction, a negative weight cycle in G becomes a positive weight cycle in G'. However, a zero weight cycle in G becomes a negative weight cycle in G'. Since the Bellman-Ford algorithm can identity negative weight cycles if one exists in G', it can be used to identify the zero weight cycles of G. Note that while the Bellman-Ford algorithm requires only polynomial computation time to find each negative weight cycle, there can be exponentially many zero weight cycles in G. Hence, the described procedure is practical only when G does not have too many zero weight cycles.

Appendix B

In a directed graph $G = (V, E)$ where no parallels edges pointing from one node to another node are allowed (e.g., the time precedence graph), a walk can be represented by a sequence of node indices $w = \{i_0, i_1, \ldots, i_m\}$ where $(i_k, i_{k+1}) \in E$ for $k = 0, 1, \ldots, m-1$. To decompose a walk w, the following "scan" operation is necessary:

$S = \text{scan}(w)$

- Initialize $S \leftarrow \emptyset$ and $k \leftarrow 0$.
- While $k < \text{length}(w)$, do
  1) If there exists r as the smallest index such that $r > k$ and $i_r = i_k$, then update $S \leftarrow S \cup \{i_k, i_{k+1}, \ldots, i_r\}$. The sequence w is also updated according to
  $$w \leftarrow \begin{cases} \{\ \} & \text{if } k = 0 \text{ and } r = m \\ \{i_0, i_1, \ldots, i_k, i_{r+1}, i_{r+2}, \ldots, i_m\} & \text{otherwise} \end{cases}.$$
  In updating w, the sub-sequence $\{i_{r+1}, \ldots, i_m\}$ is empty by convention if $r = m$.
  2) Increase $k \leftarrow k + 1$.

  End (of While)
- Update $S \leftarrow S \cup w$.

For any two nodes $u \neq v$, applying the scan operation to a walk $w = \{u = i_0, i_1, \ldots, i_m = v\}$ results in a path from u to v and a finite number of closed walks (if any). The scan operation can be applied to each closed walk and all closed walks generated subsequently. The recursive application of the scan operation eventually decomposes all closed walks into cycles in finite number of steps. To see this, each time when "children" closed walks are generated by passing a

"parent" closed walk through scan, the number of edges of the children closed walks must be smaller than that of the parent. The scan operation can be applied to each children which is not a cycle, which might generate more "grand-children" closed walks with even fewer edges. However, the recursive application of scan cannot continue indefinitely since all closed walks with two edges are cycles.

APPENDIX C

The main difficulty in the sample precedence graph construction is to guarantee that all cycles have negative weights. This is ensured by a set of specially chosen edge weights as follows: Let x_i, $i = 1,2,\ldots,n$ each be a sample of a uniform discrete random variable with values between 1 and $N(=50)$. Then, the edge weights c_{ij}'s are defined according to

$$c_{ij} = \begin{cases} x_i - x_j & \text{if } i < j, \\ x_j - x_i - z_{ij} & \text{if } i > j, \end{cases} \tag{9}$$

where z_{ij} is a sample of a uniform discrete random variable with value between 1 and $Z(=10)$. Let $(i_0, i_1, \ldots, i_m)$, with $i_0 = i_m$, denote a cycle in the precedence graph. Then there must exist a nonempty index set K such that $i_k > i_{k+1}$ for all $k \in K$. Consequently, according to (9) the weight of the cycle is

$$\sum_{j=0}^{m-1} c_{i_j i_{j+1}} - \sum_{k \in K} z_{i_k i_{k+1}} = 0 - \sum_{k \in K} z_{i_k i_{k+1}} < 0.$$

Hence, in all samples of precedence graph constructed in Section IV the negative weight cycle assumption is guaranteed.

REFERENCES

[1] K. C. Sou, J. Weimer, H. Sandberg, and K.H. Johansson, "Scheduling smart home appliances using mixed integer linear programming," in *Decision and Control and European Control Conference (CDC-ECC), 2011 50th IEEE Conference on*, dec. 2011, pp. 5144 –5149.

[2] K.C. Sou, M. Kördel, J. Wu, H. Sandberg, and K.H. Johansson, "Energy and co2 efficient scheduling of smart home appliances," in *European Control Conference*, 2013.

[3] A. Esser, A. Kamper, M. Frankje, D. Most, and O. Rentz, "Scheduling of electrical household appliances with price signals," in *Operation Research Proceedings*, 2006, pp. 253–258.

[4] T. Bapat, N. Sengupta, S. K. Ghai, V. Arya, Y. B. Shrinivasan, and D. Seetharam, "User-sensitive scheduling of home appliances," in *Proceedings of the 2nd ACM SIGCOMM workshop on Green networking*, 2011, pp. 43–48.

[5] N. Gatsis and G. Giannakis, "Residential demand response with interruptible tasks: Duality and algorithms," in *Decision and Control and European Control Conference (CDC-ECC), 2011 50th IEEE Conference on*, dec. 2011.

[6] P. Du and N. Lu, "Appliance commitment for household load scheduling," *Smart Grid, IEEE Transactions on*, vol. 2, no. 2, pp. 411–419, 2011.

[7] G. Xiong, C. Chen, S. Kishore, and A. Yener, "Smart (in-home) power scheduling for demand response on the smart grid," in *Innovative Smart Grid Technologies (ISGT), 2011 IEEE PES*, 2011, pp. 1–7.

[8] T. Ibaraki and N. Katoh, *Resource Allocation Problems: Algorithmic Approaches, no. 4 in Foundations of Computing Series*. The MIT Press, Cambridge, MA, 1988.

[9] K. C. Sou, H. Sandberg, and K. H. Johansson, "Nonserial dynamic programming with applications in smart home appliances scheduling – part II: Nonserial dynamic programming," preprint, available at the first author's homepage, 2013. [Online]. Available: http://www.math.chalmers.se/~cheong/NSDPpart2.pdf

[10] T. E. Uher, *Programming and Scheduling Techniques*. UNSW Press, 2003.

[11] Z. Mahjoub and F. Karoui-Sahtout, "Parallel algorithms for redundant precedence relations elimination in task systems," *Parallel Computing*, vol. 17, pp. 471 – 481, 1991.

[12] B. Pradeep and C. S. R. Murthy, "A constant time algorithm for redundancy elimination in task graphs on processor arrays with reconfigurable bus systems," *Parallel Processing Letters*, vol. 03, no. 02, pp. 171–177, 1993.

[13] Y. Chang, J. Pinilla, J. Kao, J. Dong, K. Ramaswami, and F. Prinz, "Automated layer decomposition for additive/subtractive solid freeform fabrication," in *Proceedings of the Solid Freeform Fabrication Symposium, The University of Texas at Austin*, 1999, pp. 111–120.

[14] J. M. Pinilla, J.-H. Kao, and F. Prinz, "Compact graph representation for solid freeform fabrication (sff)," *Journal of Manufacturing Systems*, vol. 19, no. 5, pp. 341 – 354, 2001.

[15] J. Tsitsiklis and D. Bertsimas, *Introduction to Linear Optimization*. Athena Scientific, 1997.

[16] D. Gleich, "Contents matlab bgl v4.0," 2006.