An Agent-Based Extensible Climate Control System for Sustainable Greenhouse Production

Jan Corfixen Sørensen[1], Bo Nørregaard Jørgensen[1], Mark Klein[2], and Yves Demazeau[3]

[1] The Maersk Mc-Kinney Moller Institute, University of Southern Denmark, Campusvej 55, 5230 Odense M, Denmark
{jcs,bnj}@mmmi.sdu.dk
HTTP://www.mmmi.sdu.dk

[2] Center for Collective Intelligence, MIT Sloan School of Management Five Cambridge Center NE25-754, Cambridge MA 02139, US
m_klein@mit.edu
HTTP://cci.mit.edu/klein

[3] Laboratoire d'Informatique de Grenoble CNRS, BP 53 38041 Grenoble, France
Yves.Demazeau@imag.fr
HTTP://membres-lig.imag.fr/demazeau

Abstract. The slow adoption pace of new control strategies for sustainable greenhouse climate control by industrial growers, is mainly due to the complexity of identifying and resolving potentially conflicting climate control requirements. In this paper, we present a multi-agent-based climate control system that allows new control strategies to be adopted without any need to identify or resolve conflicts beforehand. This is achieved by representing the climate control requirements as separate agents. Identifying and solving conflicts then becomes a negotiation problem among agents sharing the same controlled environment. Negotiation is done using a novel multi-objective negotiation protocol that uses a generic algorithm to find an optimized solution within the search space. The multi-agent-based control system has been empirically evaluated in an ornamental floriculture research facility in Denmark. The evaluation showed that it is realistic to implement the climate control requirements as individual agents, thereby opening greenhouse climate control systems for integration of independently produced control strategies.

Keywords: Feature interaction, Negotiation, Resource contention.

1 Introduction

In Northern Europe, the production of ornamental pot plants depends on greenhouses equipped with artificial heating and lighting systems, as heat and light are here restricting climatic factors for growth. To make this production ecologically and economically sustainable there is a critical need for energy-efficient climate control strategies that do not compromise product quality. Due to its urgency this issue has attracted the attention of an increasing number of researchers during the past decade and several research projects have produced promising control strategies [1,2,3,4,5]. Contrary to all expectations the industrial adoption pace of those control strategies has been very slow. The main reason seems to be the intrinsic complexity of combining climate control requirements of control strategies originating from independent research projects, as the optimal greenhouse climate prescribed by the climate control requirements of one control strategy may differ from or even conflict with the climates prescribed by others. Basically, independence of work implies that control strategies may have different requirements for the same climatic growth factors at the same time. If the span between requirements for a shared growth factor is narrow, it may be possible to combine the corresponding control strategies. However, if the span is broad, the requirements of the control strategies are most likely conflicting, making their combination infeasible.

Generally, when combining independently-procured control strategies they become implicitly interrelated through sharing of resources in their environment, which is recognized as a typical cause of conflicts among requirements of individual program features [6,7,8]. This is referred to as the feature interaction problem [9]. *Feature interactions* occur whenever the modification or addition of a system feature interferes with the correctness of other system features. In the worst-case scenario, such feature interactions can compromise the correctness of the overall system behavior and cause unexpected runtime faults that may lead to system failure. Hence, creation of an independently extensible greenhouse control system, in which feature interactions do not happen, requires an implementation approach that is capable of coordinating the effects of independent control strategies on shared growth factors in such a way that the requirements of all control strategies are satisfied. Multi-agent systems provide an implementation approach that can support independent extensibility by modeling the units of composition as autonomous agents. Using multi-agent systems allow us to achieve separation of specifications by implementing each climate control requirement as an agent. Hence, prevention of feature interactions becomes a question of coordinating the agents' actions on the controlled environment such that the goal of each agent is satisfied without compromising the goals of others. As the desired effects of the agents' goals for the shared growth factors are described by non-linear multi-variable functions, the coordination of the agents' actions on the environment constitutes an open-ended multi-objective negotiation problem with non-linear utility functions. In this paper, we propose a multi-agent system that allows independent extensibility of greenhouse climate control systems by using a novel multi-objective negotiation protocol to find an optimized greenhouse climate which satisfies the requirements of all control strategies.

The paper is organized as follows. Section 2 describes the typical setup for greenhouse climate control. Section 3 presents requirements for an energy-efficient production. Section 4 presents our multi-agent system. Section 5 describes the implementation of our negotiation protocol. Experimental validation of our approach is presented in Section 6 using a data set obtained from a production greenhouse. Section 7 discusses related work. Section 8 highlights future work. Finally, we conclude our work in Section 9.

2 Greenhouse Climate Control Setup

In general, a greenhouse climate control system is a computer system that controls the climate-related factors for growth by sensing and manipulating the greenhouse climate through the use of sensors and actuators. Sensors are used to measure the actual levels for each of the growth factors while actuators are used to change them. Measured growth factors include temperature, light, CO_2, and humidity. Actuators include the lighting, heating, CO_2, window, curtain and irrigation subsystems of the greenhouse. The lighting subsystem adds supplemental light when the present natural light level is insufficient for sustaining the required plant growth. The heating subsystem is used to maintain the right temperature during cold periods. The CO_2 subsystem doses CO_2 to increase the photosynthesis efficiency of the plants. The window subsystem lowers the temperature and the humidity by opening the windows. Similarly, the curtain subsystem can lower the temperature by shading the plants. Furthermore, curtains are used at night during the winter season to isolate the greenhouse. The irrigation system is responsible for watering the plants. The control of these subsystems is linked to sensor readings through the combined control strategy of the greenhouse's climate control system. The combined control strategy prescribes what is considered to be the optimal greenhouse climate in terms of *temperature*, *light*, *CO_2* and *humidity levels*. The combined control strategy must also coordinate the subsystems of the climate control system, such that no unwanted interactions emerges. For instance, the CO_2 subsystem should not dose CO_2 while the windows are open, as the CO_2 will simply diffuse before it is absorbed by the plants' photosynthesis process. Hence, coordination of the subsystems is vital to ensure the correctness of the climate control. Figure 1 depicts the actual setup of our climate greenhouse control system.

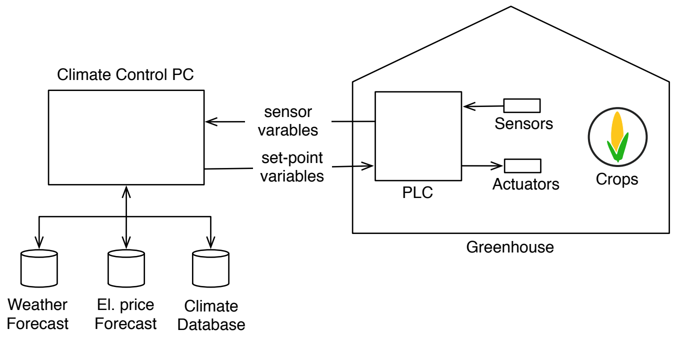


Fig. 1. Climate Control System Overview

Our climate control system consists of a climate control PC that is connected to three database servers for accessing weather forecasts, electricity prices and historical climate data. Additionally, the climate control PC is connected to a Programmable Logic Controller (PLC) that is physically connected to the sensors and actuators. The role of the PLC is to convert signals from the sensors into sensor variables that can be read by the control strategy running on the climate control PC. Furthermore, the PLC converts set-point variables from the climate control PC into signals that can be sent to the physical actuators.

3 Requirements in Greenhouse Climate Control

Requirements for energy-efficient greenhouse production are closely related to the use of artificial heating and lighting systems, as heat and light constitute the restricting climatic growth factors from late autumn to early spring. The IntelliGrow climate control system introduced in [1] provides a control strategy for efficiently reducing the amount of energy required for heating. The system uses a mathematical model of the plants' photosynthesis process to optimize the temperature and CO_2 levels according to the actual light level in the greenhouse. Compared to traditional climate control strategies, which use constant temperature settings for predefined time periods, IntelliGrow does not waste energy on unnecessary heating under low light conditions. A similar approach was demonstrated to reduce energy consumption for use of supplemental light in [10]. Here the same photosynthesis model is used together with weather forecasts and electricity prices to compute a light plan that optimizes the photosynthesis gain with respect to energy consumption and electricity costs. The result is a growth and energy-related control strategy that does not compromise product quality. Other issues may also influence the use of artificial lighting. For instance, during winter working light is required in the early morning and late afternoon. By inspecting the proposed approaches and work-related restrictions we can now identify the following requirements:

[$r_{LightHour}$] is a requirement to ensure light to be switched on at specific hours. For example, artificial lighting could be used as working light during dark winter mornings.

[$r_{DarkHour}$] is a requirement to ensure a fixed number of dark hours. For example, light can be forced off during specific dark hours. Most cultivars require at least two hours of total darkness during the night.

[$r_{OptimalPhotosynthesis}$] is a requirement to ensure that the temperature and CO_2 levels are optimal with respect to the actual light level in the greenhouse.

[$r_{GrowthGoal}$] is a requirement to ensure that a specific growth goal expressed as an accumulated photosynthesis gain is achieved.

[$r_{MinLightPrice}$] is a requirement that minimizes the price of the light plan for a given period based on forecasted electricity prices.

4 Multi-Agent-Based Control System

The premise for our approach to the feature interaction problem is that the feature interaction problem can be perceived as a search problem. In the search problem, each climate control requirement is represented by an agent and the feature interactions are represented as conflicts between the agents' goals. The objectives are defined in terms of actuator set points that each agent needs to agree upon to fulfil the climate control requirements of the combined control strategies. The resolution of the feature interactions is accomplished by a *trusted negotiator* through negotiation with the agents over the set of objectives.

An agent has beliefs about the environment in terms of sensor inputs, desires in terms of goals that the agent attempts to fulfil to accomplish the climate control requirement it represents, and intentions to accept or reject the proposed options from the negotiator based on its internal goals. Agent goals are specified in terms of the growth factors that is indirectly influenced by the sensors and actuators through the environment. Therefore, an agent can only be added if the corresponding sensors and actuators are present in the control system. For example, if an agent's goals is specified in terms of light levels, then a light sensor needs to be represented in the control system. Likewise, sensors and actuators can only be removed from the control system, if none of the agents' goals indirectly depends on them. Typically, the set of sensors and actuators stay constant over time and are often added/removed only when the physical greenhouse configuration is changed.

The trusted negotiator manages the negotiation protocol and creates options for a consensus solution that has potential to fulfil the climate control requirements represented by the agents. The negotiator starts a negotiation for each control cycle by suggesting a number of options that have potentials to become a solution. An option is the negotiator's assignment of values for actuator set points based on a given set of sensor-input values. The negotiation process is governed by a protocol which specifies the interaction between the negotiator and the agents, ensuring that only allowed messages are sent and that the messages are sent in the right order. The protocol constitutes four different messages: *option*, *accept*, *satisfy* and *alert* messages. The negotiator asks each agent if the proposed option is acceptable. Each agent evaluates each proposed option against its internal goals and responds back to the negotiator with an accept message if the option is within the boundary conditions of its goals. Agents that cannot accept a proposed option responds with a *reject* message. Additionally, if an agent accepts the proposed option, the negotiator asks the agent how well its goals are met. The agent answers back with a satisfy message that specifies a value for how well the goals of the agent are satisfied based on evaluation of the proposed option. A satisfy message is expressed in terms of an objective criteria value in the interval $[0; 1]$. A value of zero means that the goals are fully satisfied by the option while a value of one means the goals are not well satisfied by the proposed option. The negotiator computes the overall *satisfaction degree* for the proposed option given the accept and satisfy messages from the agents. The trusted negotiator repeatedly generates and proposes options to the agents until the negotiation process terminates. The negotiation process terminates after a specified amount of time depending on the time interval between control cycles. At termination of the negotiation process, the negotiator selects the option with the best satisfaction degree as the solution that can be effectuated by the control system.

In cases where no acceptable solution can be found, the system will continue to run, but the negotiator will send an *alert* message to the user of the system that explains which agents are in conflict with each other. Based on information about the conflicting agents and information about which set points they share, the domain expert user of the system can make an informed adjustment of the conflicting goals to resolve the conflicts. *Explanation* of conflicts is an important feature of our approach as it may be impossible to find solutions in situations where conflicts emerge as a consequence of conflicting requirements. In such situations the goals of the agents need to be relaxed by an informed decision made by the user of the system.

5 Negotiation-Based Coordination

Most approaches to multi-objective problems are based on an optimization process. Focus on optimization requires definition of a global optimization criterion that the solution should get close to. In our open-ended multi-objective control domain, it is difficult to establish such an optimization criterion because information is limited. By definition, an extensible system is never complete and as a consequence optimization approaches that require complete information cannot be applied to independently extensible systems [11]. In other words, the agents can only know about their shared environment and has no information about the other agents. For that reason, agents will act with bounded rationality rather than with economical rationality in terms of utility. In open-ended environments with limited information, it is not clear what defines the best solution. The main limitation of information in our domain is how the plants will be affected by control parameters, since there are no complete integrated plant models available. That is, models only represent parts of the environment. Even if an optimization criterion can be defined, it is not certain that a solution can be found because of limited information. For example, the size of search space for an artificial light-control system with a 24-hour light-plan output would be 2^{24}, assuming light is turned on/off once per hour. Thus, the size of search space explodes as the number of control set points increases. Last but not least, if there is an optimal solution, it is uncertain whether it can be found within reasonable time with current computing power [12].

Our approach, therefore, focuses on satisficing rather than optimization to overcome the problem of limited information [13,14]. Instead of defining a global optimization criterion, we define a *satisficing criterion* that a given solution should fulfil. In other words, a good enough solution is a solution that satisfies the boundary constraints of the agents' goals. A satisficing solution may or may not be an optimal economic solution. The negotiation process is implemented as a genetic algorithm that is executed for every control cycle of the control system. The process is described in Pseudocode 1 and includes four phases: 1) create initial options, 2) evaluate each option against all requirements, 3) create new options based on evaluation, and 4) select best option as a solution when all negotiation rounds have been executed.

First, *phase 1* of the negotiation process is started by the negotiator that creates a random set of options *optionSet* that are open for negotiation, see line 2 of Pseudocode 1. For example, the negotiator creates a number of different random light plans. The size of the option set is determined by population size in the genetic algorithm that is found by experimental fine-tuning.

Pseudocode 1. Negotiation process

```
1: {Phase 1}
2: $optionSet = createInitialOptions(system)$
3: for all $round = 0 \rightarrow maxRounds$ do
4:    {Phase 2}
5:    for all $option \in optionSet$ do
6:       $option.satisfySum = 0$
7:       $option.accepted = 0$
8:       for all $agent \in agentSet$ do
9:          if $agent.accept(option)$ then
10:            $option.satisfySum = option.satisfySum + agent.satisfy(option)$
11:         else
12:            $option.accepted = 1$
13:         end if
14:      end for
15:      $option.fitness = option.accepted + option.satisfySum/size(agentSet)$
16:   end for
17:   {Phase 3}
18:   $sort(optionSet)$
19:   $optionSet = createNewOptions(optionSet)$
20: end for
21: {Phase 4}
22: $solution = selectBestSolution(optionSet)$
23: if $\neg solution.accepted$ then
24:    $sendAlert(solution)$
25: end if
```

Phase 2 starts the first negotiation round given the initial option set as a start condition. The negotiator asks each participating agent if it can accept the option from the option set. Each agent evaluates the proposed option based on its beliefs and goals and responds back with an accept message if it can accept. Formally, each requirement is represented by an agent. For example, the requirement $r_{DarkHour}$ is represented by the agent $a_{DarkHour}$ with its goal to ensure that light is turned off for a specified number of hours. The goal $goal_{DarkHour}$ is expressed as an inequality goal constraint over the agent's dark-hour plan and

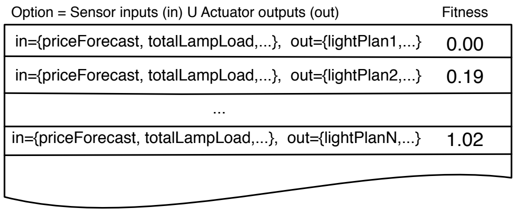
Pseudocode 2. $a_{DarkHour}.accept(option)$

```
1: for all hour ∈ option.lightPlan do
2:    if option.fixedPlan[hour] ≠ option.lightPlan[hour] then
3:       return false
4:    else
5:       return true
6:    end if
7: end for
```

Pseudocode 3. $agent_{MinLightPrice}.satisfy(option)$

```
1: for all hour ∈ option.lightPlan do
2:    if isOn(hour, option.lightPlan) then
3:       $consumedEnergy = option.totalLampLoad \times seconds(hour)$;
4:       $energyCost = option.priceForecast[hour] \times consumedEnergy$;
5:       $energyCostSum = energyCostSum + energyCost$;
6:    end if
7: end for
8: return $objectiveCriteria(energyCostSum)$;
```

the proposed light plan from the negotiator. The intention of agent $a_{DarkHour}$ is to respond with an accept message if the goal constraint is not broken for all hours in the light plan. If an agent accepts an option, the negotiator asks the agent for a satisfy message. The satisfiability value returned by the agent expresses how well the proposed options satisfy the agent's goals. For example, the requirement $r_{MinLightPrice}$ is represented by agent $a_{MinLightPrice}$ with the goal $g_{MinLightPrice}$. The goal $g_{MinLightPrice}$ is to minimize the cost of the proposed light plan as much as possible, see Pseudocode 3. The intention of the agent is to accept a proposed light plan and to send a satisfy message that expresses how well the cost of the light plan was minimized. The two different agents represent two different kinds of requirements. Agent $a_{DarkHour}$ represents a requirement that is fully satisfied if the agent's goal constraint is not broken by the proposed option. Oppositely, agent $a_{MinLightPrice}$ represents an optimization requirement to the proposed option. For example, using accept and satisfy messages we can represent constraint-based requirements that are expressed as goal constraints and optimization requirements that are expressed as an utility value. Based on the received satisfy messages, the negotiator calculates a satisfy sum $options.satisfySum$ for the option, see line 10 of Pseudocode 1. If the agent responds with a reject message, the negotiator marks the options as being not accepted (value 1), see line 12 of Pseudocode 1. When all agents have responded on the proposed options, the negotiator marks the option with a fitness value calculated based on option acceptance and the option's satisfy sum, see line 15 of Pseudocode 1. The lower the fitness value is, the better the option is. The marking of options continues until all options in the option set *optionSet* have been marked with a fitness value.



Option = Sensor inputs (in) U Actuator outputs (out)	Fitness
in={priceForecast, totalLampLoad,...}, out={lightPlan1,...}	0.00
in={priceForecast, totalLampLoad,...}, out={lightPlan2,...}	0.19
...	
in={priceForecast, totalLampLoad,...}, out={lightPlanN,...}	1.02

Fig. 2. An option set sorted according to option fitness

In *phase 3*, the set of options are now sorted according to their fitness. The result is an ordered set of options. Options that are accepted by most agents and have the best average satisfiability appear first in the set, see Figure 2. The negotiator examines the ordered option set and replaces the worst half of the set with new options. The new options are generated by either mutation or crossover of options, selected randomly across the entire option set. The $isMutation()$ is a function that randomly returns true or false for whether or not an option i should be mutated or replaced by a crossover result, see line 2 of Pseudocode 4. Each *mutation operator* is domain-specific and allows the developer to specify the allowed variability of the set-point values. For example, the mutation function for a light plan is specified in Pseudocode 5. Domain-specific mutation functions improve the performance of the genetic algorithm as they make the search more directed and avoid mutating options into options that are unrealistic; for example, mutating a temperature set point into a value that is outside the allowed range of temperatures. A domain-specific mutation function for a temperature avoids that problem by specifying mutation within a specific temperature range. The *crossover operator* is generic and is randomly applied to half of the outputs for the selected option to be crossed, see Pseudocode 6. The random selection together with elimination of the worst individuals reduce the chance of ending up with a suboptimal solution since all suitable individuals have a chance to mutate and reproduce. The updated options become the new input for the next negotiation round (generation). The negotiation process continues for a fixed number of negotiation rounds.

Pseudocode 4. $createNewOptions(optionSet)$, see line 19 of Pseudocode 1

```
Require: optionSet exists
Ensure: new optionSet
 1: for $i = size(optionSet)/2 \rightarrow size(optionSet)$ do
 2:    if $isMutation()$ then
 3:       $optionSet[i] = mutate(random(optionSet))$
 4:    else
 5:       $optionSet[i] = crossover(random(optionSet), random(optionSet))$
 6:    end if
 7: end for
```

Pseudocode 5. *mutate*(*option*), see line 3 of Pseudocode 4

```
for $i = 0 \rightarrow size(option.lightPlan)/4$ do
    $idx = randomIndex(option.lightPlan)$
    $option.lightPlan[idx] = \neg option.lightPlan[idx]$
end for
return $option.lightPlan$;
```

Pseudocode 6. *crossover*(*optionA*, *optionB*), see line 5 of Pseudocode 4

```
for $i = 0 \rightarrow size(optionA.out)/2$ do
    $j = removeRandom(optionA.out)$
    $optionA.out[j] = optionB.out[j]$
end for
return $optionA$
```

The end of the negotiation rounds marks the start of phase 4, where the solution of the negotiation rounds is selected as the option with the best fitness. If the selected solution is marked as not accepted, it means that not all agents accepted the solution and that there are conflicts between the goals of the agents. The negotiator generates an informed alert message that explains which agents are in conflict with each other and which sensor input and actuator set points they share. The alert message is sent to the user of the system that can then make an informed decision to relax one or more of the goals of the conflicting agents. The relaxation of goals will be taken into consideration in the next control cycle when a new negotiation process is triggered. Finally, the set points from the selected solution are sent to the PLC and effectuated by the subsystems' actuators.

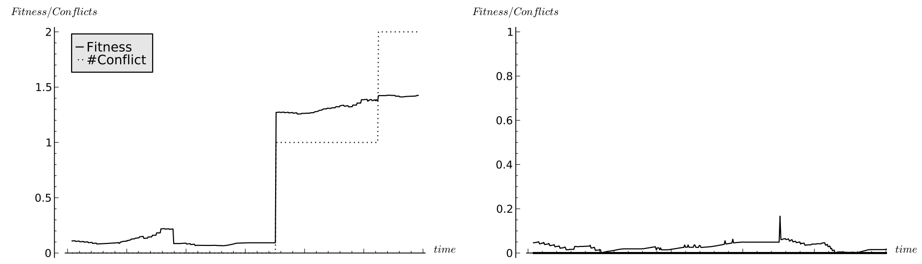
6 Experimental Validation

The evaluation of our solution is based on real data from an experiment conducted in the period from 12th October to 9th November 2009 with a total of 300 cuttings of *Chrysanthemum Morifolium*. The motivation of the experiment was to identify how irregular light periods affect the plant growth of *Chrysanthemum Morifolium*. The results confirm that climate-control strategies based on hourly changes in electricity prices can have an economical value for the production of pot plants [10]. The light strategy to generate irregular light periods was provided by a system (DynaLight earlier also known as Climate Monitor) that implemented the requirements $r_{GrowthGoal}$, $r_{MinLightPrice}$, $r_{LightHour}$, $r_{DarkHour}$ and $r_{OptimalPhotosynthesis}$ described in Section 3.

DynaLight represents a centralized system with all requirements implemented in one specification without any negotiation mechanism to coordinate and explain possible conflicts between the requirements [15]. Contrary, our approach represents a system with all requirements implemented as independent separate agents with a negotiation mechanism to manage and explain conflicts between the agents. Since the requirements are similar in the two systems, they can be represented by the same set of agents and it is possible to compare the fitness of the solutions and the number of conflicts generated. For that reason, the data generated by DynaLight is ideal for an experimental evaluation of our multi-agent-based control system. The purpose of the experimental evaluation is to compare the fitness of the solutions from DynaLight with the fitness of the solutions from our approach. Additionally, the experiment validates the ability of our approach to manage and explain conflicts compared to DynaLight. The experimental setting includes two experiments. The first experiment evaluates the fitness and the number of conflicts of the solutions found by DynaLight. That is, the agents only represent the requirements from DynaLight and are used to evaluate fitness and conflicts of the set-point values that were generated by DynaLight. The second experiment evaluates the fitness and number of conflicts for the same set of agents but using sensor inputs from the earlier experiment. That is, the set points are determined by the negotiation process between the agents and the negotiator. Both experiments share the same configuration from the experiment (60% NB) described in [10]. In summary, the photosynthesis optimization is 60%, the lamp intensity is 60 $\mu mol/m^2s$, the photosynthesis growth goal is 268 $mmol/m^2s^1$, the dark-hour period is 5-8pm and the control-cycle interval is 10 minutes. The time zone for the experiment is GMT+1 at wintertime. Furthermore, the total connected lamp load is 7 KW, i.e., the installed lamp effect per square meter is 70 W and the size of the greenhouse is 100 m^2. In the second experiment, the size of the option set is configured to be 500 and the number of negotiation rounds were set to 1000.

Figure 3a illustrates the fitness and the number of conflicts for the DynaLight experiment. The first conflicts start emerging midnight October 21 because artificial light is proposed to be turned on at 8 pm causing violation of the requirement $r_{DarkHour}$ because the hour at 8 pm is specified as a dark hour. The requirement $r_{DarkHour}$ persists to be compromised from midnight until the end of the experiment due to the lack of a mechanism for handling the conflict automatically. One explanation for the conflict is that the climate computer, which DynaLight interfaces, was reconfigured by accident. The second conflict occur early morning October 22 as a consequence of a proposed light plan that does not fulfil the specified photosynthesis growth goal. The proposed light plan causing the second conflict, has a total photosynthesis contribution of 110 $mmol/m^2s$. The achieved photosynthesis contribution from natural light and the light plan, at the time were the conflict emerges, is 11 $mmol/m^2s$. The estimated photosynthesis contribution from natural light for the rest of the day is 142 $mmol/m^2s$. Summed together, the proposed light plan, the achieved photosynthesis contribution from natural and artificial light, and the expected photosynthesis contribution from natural light for the rest of the day is not enough to achieve the target of a photosynthesis growth goal of 268 $mmol/m^2s$. That is, the photosynthesis sum balance calculated is $(110 + 11 + 142) - 268 = -6$ $mmol/m^2s$ which means that

[1] Photosynthesis is given as flux of Photosynthetically active radiation photons per unit area (PPFD).



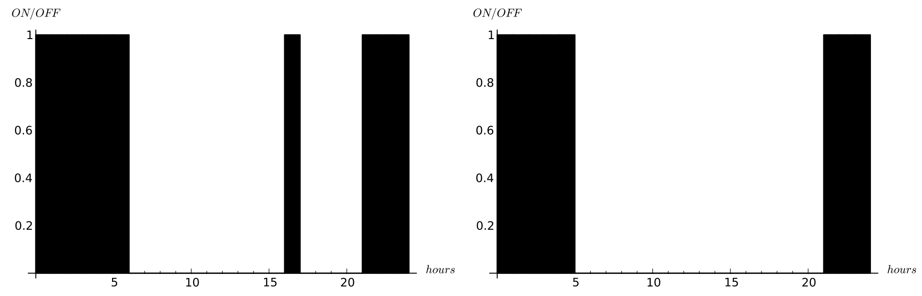
(a) DynaLight. (b) Negotiation approach.

Fig. 3. Fitness and conflicts of solutions found with and without a negotiator over a period (19-22 Oct 2009)

the agent presenting requirement $r_{GrowthGoal}$ lacked 6 $mmol/m^2s$ to reach its growth goal. Figure 3b depicts the result of our negotiation mechanism where no conflicts emerge as the negotiator finds solutions that satisfy the requirements.

The average fitness of the result from the DynaLight experiment, in the period till the first conflict occurrence, is 0.1044 and the average fitness for the entire period is 0.4488. In comparison, the fitness of the negotiated result for the same period before any conflicts is 0.0298 and the average fitness for the entire period is 0.0295. To explain the relationship between fitness and the solutions found it is relevant to investigate light plans from the two experiments before the occurrence of any conflicts. Figure 4a and 4b depict the light plans effectuated October 19 by DynaLight and our approach. The cost of the DynaLight light plan (Figure 4a) is calculated to €2.44 based on the forecasted electricity prices and the connected lamp load. Moreover, the photosynthesis contribution from the DynaLight light plan is 115 $mmol/m^2s$. Conversely, the cost of our negotiated light plan for the same period is €1.94 and the photosynthesis contribution is 92 $mmol/m^2s$. In summary, that means there is a saving of €0.50 for the day for the negotiated solution but that the photosynthesis contribution from the negotiated light plan is 23 $mmol/m^2s$ less than is gained from the DynaLight light plan.

The poor fitness of the light plan from DynaLight can be explained by understanding the way DynaLight plans ahead of time. DynaLight generates its light plan based on a daily analysis of the forecasted natural light. If the forecasted natural light is pessimistically estimated at the time DynaLight generates its light plan, the result will be a too large photosynthesis contribution from the light plan that together with the actual achieved photosynthesis contribution from natural light will exceed the specified growth goal. Exceeding the growth goal leads to a poor fitness of requirement $r_{GrowthGoal}$. Additionally, a pessimistic forecast leads to a higher cost (bad fitness of requirement $r_{MinLightPrice}$) as the photosynthesis growth goal can be achieved with less artificial light because of the higher photosynthesis contribution from the actual natural light. In contrast, our approach negotiates the light plan for each control cycle and continuously adapts the light plan according to the actual achieved photosynthesis contribution from natural light. The result is a dynamic light plan that gets very



(a) DynaLight. (b) Negotiation approach.

Fig. 4. Light plans for 24-hours found October 19 2009 with and without a negotiator

close to the specified growth goal, resulting in a better fitness of both requirement $r_{GrowthGoal}$ and $r_{MinLightPrice}$. In fact the 92 $mmol/m^2s$ is just enough to achieve the growth goal given the forecasted natural light. Last, the light plans for 19th October look very similar which could be expected considering both approaches represent the same set of requirements.

7 Related Work

Our work is inspired by [7,8,16,17] which differ from other prevalent approaches by perceiving the feature interaction problem as a resource-sharing problem rather than a feature-behavior problem. The idea behind the approaches is based on the assumption that feature interactions emerge as a consequence of features sharing resources. The argument for focusing on resources instead of feature behaviors is that resources are simpler to model and understand than feature behaviors. To manage feature interactions, the approach requires a specification based solely on knowledge about the resources.

Bisbal and Cheng contribute with a resource-oriented approach to detect and handle feature interactions in component-based software at runtime [7]. Their resource-aware specification declares the resource goals of a component and the relationship between the component and its resources. The specification is declared at design time and used by runtime techniques to address feature interactions in resource-aware systems.

Liu and Meier contribute with resource-aware contracts and address resource-based feature interactions in dynamic adaptable systems [17]. The resource specification proposed by Bisbal and Cheng is based only on fixed-capacity and varying-capacity resources and can only be applied at component level. In contrast, resource-aware contracts support both fixed-capacity, varying-capacity, exclusive and shared resources and can be applied at both component and component-assembly level. In addition, resource-aware contracts also specify the global resource constraints at system level.

Zambrano et al. focus on aspect interactions and use metadata annotations to specify the resource requirements of each aspect [16]. Zambrano et al. provide a detection and a resolution strategy that can detect and avoid feature interaction in resource-aware systems, without compromising obliviousness of the aspects. The resource specification is declared as metadata on the aspects at design time in the form of semantic annotations. The interactions between aspects are avoided at runtime by a coordinator aspect. The coordinator aspect is augmented with a list of user-defined conflict situations declared at design time as conflict rules. A conflict rule expresses the resource conditions that should be avoided by corresponding corrective actions. Affected resources are asserted against a coordinator rule engine to detect conflict conditions. To avoid interactions, the coordinator aspect can deactivate the conflicting aspects as declared in the action part of the triggered conflict rule.

The approaches suggested by Bisbal, Cheng, Liu, Meier and Zambrano et al. are based on formal specifications/analysis of the resources shared between the features. The analysis of the resources can be accomplished at design time and applied to runtime approaches to enhance detection and resolution of feature interactions. Feature interactions are perceived as violations of the feature requirements as a consequence of wrong assumptions about the shared resources. The resource specifications are described independently of the features and support development of features by third-party vendors. In case no solutions can be found to resolve the feature interaction, the mentioned approaches do not support that the user can redefine the requirements of the conflicting features to find alternative solutions. Redefinition of conflicting feature requirements to resolve interactions requires explanation of what caused the feature interaction. To our knowledge none of the approaches support such detailed *explanation* of the cause of feature interactions.

8 Future Work

Explanation of runtime feature interactions requires that the agents are kept separate and represent requirements as perceived by the user of the system. That is, if the requirements are not understood by the user then it is difficult to explain the conflicts as they are expressed in terms of the requirements. For that reason, future work should incorporate a description language that can support the creation of more user friendly alert messages. The explanation of feature interactions should be improved by supporting visual views that explain the interactions as a resource sharing problem among agents. One suggestion for a visual view, would be to visualize the shared outputs over time using color scales corresponding to the number of conflicting agents. For example, if one agent can't accept the negotiated solution because the output is negatively influenced by other agents, then that shared output should be colored red. The conflicting resource should be colored more dark red when the number of conflicting agents increases. Additionally, the color-view of the resources can be combined with a color-view of the agents showing which agents are causing the resource conflicts. Conflicting agents should be colored red if they didn't accept any of the proposed options, otherwise the agents should be colored according to their satisfaction degree. Additionally, the overall performance of the system should be visualized using a colored graph-view illustrating the satisfaction degree of the found solutions compared to the satisfaction of each of the agents. Finally, the approach needs to be more thoroughly tested. It is obvious to compare the approach with our earlier proposed counter-proposal negotiation approach to evaluate how well the approach performs [18]. An other aspect that should be evaluated, is how well the approach supports explanation compared to other negotiation approaches.

9 Conclusion

In this paper, we presented a novel multi-agent-based approach to control system engineering that frees developers from identifying and resolving interactions among control strategies before they can be deployed as part of the same system. Our approach achieves this by implementing each control requirement as a separate agent which then participates in the negotiation of acceptable values for the control set points. Through empirical evaluation in an ornamental floriculture research facility, we have shown that it is realistic to implement independent climate control requirements as individual agents, thereby opening greenhouse climate control systems for integration of independently produced control strategies.

References

1. Aaslyng, J., Lund, J., Ehler, N., Rosenqvist, E.: IntelliGrow: a greenhouse component-based climate control system. Environmental Modelling & Software 18(7), 657–666 (2003)
2. Markvart, J., Kalita, S., Jørgensen, B.N., Aaslyng, J.M., Ottosen, C.O.: IntelliGrow 2.0 - a greenhouse component-based climate control system. In: Proceedings of the International Symposium on High Technology for Greenhouse System Management (2007)
3. Körner, O., Challa, H.: Temperature integration and process-based humidity control in chrysanthemum. Computers and Electronics in Agriculture 43, 1–21 (2004)
4. Körner, O., Andreassen, A.U., Aaslyng, J.M.: Simulating dynamic control of supplementary lighting. Acta Horticulturae 711, 151–156 (2006)
5. Körner, O., Aaslyng, J.M., Andreassen, A.U., Holst, N.: Microclimate prediction for dynamic greenhouse climate control. HortScience 42(2), 272–279 (2007)
6. Armstrong, N., Robin, L., Bashar, N.: Feature Interaction as a Context Sharing Problem. In: Feature Interactions in Software and Communication Systems X (2009)
7. Bisbal, J., Cheng, B.H.C.: Resource-based approach to feature interaction in adaptive software. In: WOSS 2004: Proceedings of the 1st ACM SIGSOFT Workshop on Self-Managed Systems, pp. 23–27. ACM, New York (2004)
8. Metzger, A.: Feature interactions in embedded control systems. Computer Networks 45(5), 625–644 (2004)

9. Calder, M., Kolberg, M., Magill, E.H., Marganiec, S.R.: Feature interaction: a critical review and considered forecast. Comput. Netw. 41(1), 115–141 (2003)
10. Kjaer, K.H., Ottosen, C.O.: Growth of Chrysanthemum in Response to Supplemental Light Provided by IrregularLight Breaks during the Night. Journal of the American Society for Horticultural Science 136, 3–9 (2011)
11. Szyperski, C.: Independently Extensible Systems - Software Engineering Potential and Challenges. In: Proceedings of the 19th Australasian Computer Science Conference (1996)
12. Goodrich, M., Stirling, W., Frost, R.: A satisficing approach to intelligent control of nonlinear systems. In: 1996 IEEE International Symposium on Intelligent Control, pp. 248–252. IEEE (September 1996)
13. Simon, H.A.: A Behavioral Model of Rational Choice. The Quarterly Journal of Economics 69(1), 99–118 (1955)
14. Simon, H.: Optimal problem-solving search: All-or-none solutions. Artificial Intelligence 6(3), 235–247 (1975)
15. Mærsk-Møller, H.M., Jørgensen, B.N.: A Software Product Line for Energy-Efficient Control of Supplementary Lighting in Greenhouses. In: The International Conference on Green Computing (2011)
16. Zambrano, A., Vera, T., Gordillo, S.E.: Solving Aspectual Semantic Conflicts in Resource Aware Systems. In: RAM-SE, pp. 79–88 (2006)
17. Liu, Y., Meier, R.: Resource-Aware Contracts for Addressing Feature Interaction in Dynamic Adaptive Systems. In: 2009 Fifth International Conference on Autonomic and Autonomous Systems. IEEE, Los Alamitos (2009)
18. Sørensen, J.C., Jørgensen, B.N.: Counter-proposal: A Multi-Agent Negotiation Protocol for Resolving Resource Contention in Open Control Systems. In: The 9th International Conference on Autonomous Agents and Multiagent Systems - Workshop 1 Agent Communication (2009)