We are IntechOpen, the world's leading publisher of Open Access books Built by scientists, for scientists

4,800
Open access books available

122,000
International authors and editors

135M
Downloads

Our authors are among the

154
Countries delivered to

TOP 1%
most cited scientists

12.2%
Contributors from top 500 universities

CLARIVATE ANALYTICS BOOK CITATION INDEX INDEXED

WEB OF SCIENCE™

Selection of our books indexed in the Book Citation Index in Web of Science™ Core Collection (BKCI)

Interested in publishing with us?
Contact book.department@intechopen.com

Numbers displayed above are based on latest data collected.
For more information visit www.intechopen.com

5

Effective Planning for Conflicting Situations for Ubiquitous Sensor Network Environments

Toshiharu Sugawara[1] Satoshi Kurihara[2] Toshio Hirotsu[3]
Kensuke Fukuda[4] and Toshihiro Takada[5]
[1]*Waseda University,*
[2]*Osaka University,*
[3]*Hosei University,*
[4]*National Institute of Informatics,*
[5]*NTT Communication Science Laboratories,*
Japan

1. Introduction

Applications of sensor networks and ubiquitous computing have received attention. They can provide many kinds of important services for supporting daily and social activities in home, schools, offices and public spaces in the future (Kurihara, 2008). However, to realize these kinds of applications, a number of new technologies in AI and multi-agent systems (MAS) are also required because many devices and control programs are concurrently work to achieve their goals in cooperation with other ones. These works arise according to the human requirements based on their individual activities. In order to achieve these required goals, each agent has to create the plan (means-end analysis) and then performs it. However, the plan often conflict with those that are being created, already being scheduled, and executed by other agents because of the limited resources. Furthermore, since the human's activities are usually real-time with deadline, the agent must also be able to complete its planning and resolution of these conflicts within a reasonable time to have an acceptable quality plan. This means that both efficient planning and sophisticated conflict resolution are strongly required.

We adopt hierarchical planning (for example, see (Erol & Nau, 1994; Sacerdoti, 1974) using the decision-theoretic planning approach (Goldwin, & Simmons, 1998) for efficient planning but it is not trivial to apply hierarchical planning to MAS. In hierarchical planning, appropriate (abstract) plans are selected level by level to maximize the utility $U(p)$, where p is the expected final plan comprising a sequence of primitive actions. However, in the MAS context, conflicts between agents affect the efficiency and quality of resulting plans. When a conflict is found at lower levels, an additional sophisticated process for avoiding it (*conflict resolution*) must be invoked and some extra actions (such as waiting for synchronization and detouring) may have to be added to the plan. The conflict resolution process may become costly or fail. Even a single conflict, if it is difficult to resolve, will result in a plan with

considerably lower quality. As a result, in multi-agent systems, the second- or third-best plans may result in better overall performance.

The objective of our research is to enable agents, using reinforcement learning, to predict which tasks in an abstract plan will conflict with other agents' plans at a lower level with higher probability and either involve a costly conflict resolution process and/or result in a low-quality plan after it has been resolved. We emphasize that the appearances of conflicts strongly depend on the resource structures of the environments of the sensor-network applications. This suggests that the learning is mandatory.

Our basic idea is threefold, *conflict patterns*, *screening level* and *conflict discount*. First, we will introduce *conflict patterns* (*CP*) at a certain abstract level called the *screening level* (*SL*). The screening level is a one of intermediary level of the hierarchical model at which the conflicts of generating plans are predicted. The possible conflicts are stored as conflict patterns to specify the situations where conflicts will occur with high probabilities if the agents refine the current plan to the lower levels. The *conflict discount* is a negative utility that cumulatively predicts the probability of conflicts in the subsequent refinement process, the cost of resolutions, and the quality/performance of the resulting plans on the basis of CPs in the plans at the screening level and past experience. The conflict discount is calculated and updated by using statistically learned expected values or by reinforcement learning, so that the agents select the appropriate refinement at the SL.

Note that we assume that the initial utility is good for selecting plans for single-agent cases. This utility may lead to acceptable but minimum quality plans after conflict resolution in the MAS context. Thus, agents learn the conflict discount appropriate for the environment in order to select better SL plans.

In this chapter, we formally define conflict patterns and discuss the estimation of their conflict discounts. We then introduce the notion of sub-conflict patterns for avoiding redundant calculations of conflict discounts and reducing memory space. We also clarify the distributed version of the planning framework with our conflict estimation, which is an extension of that in (Sugawara et al., 2005). Then we present an experimental evaluation of the efficiency of plans generated by our method for a simulated laboratory room. This chapter is organized as follows: First, we discuss the issue addressed here and the planning framework used in our application systems. We then explain the process of conflict detection and resolution. Following that, we introduce the use of conflict patterns to classify situations involving conflicts with other agents' plans. Then, the experimental results to evaluate our approach are presented. We show that our proposed planning strategy makes agent's planning more efficient in the situation where conflicts are predicted. Finally, we cover related work and offer some concluding remarks.

2. Conflict estimation in hierarchical planning

In hierarchical planning, plans are generated using an abstract hierarchy of the domain model, which includes tasks and resources in an abstract form. Initial states and goals are first described in the most abstract model, and a number of task sequences are generated to achieve these goals. One of the sequences is then selected according to a particular planning

strategy (A utility is used in the case of the decision-theoretic planning.[1]), and each task in the sequence is further refined into task sequences in the less-abstract model. This refine-and-select process is iterated until all tasks have been refined to primitive tasks in the lowest model. In general, while abstract (higher-level) models are simple and thus do not contain complete information, they are appropriate for understanding the global and long-term picture of activities. Naturally, the lower-layer models are more informative and complicated, so they are used for detailed descriptions of local and sectional plans.

Let's consider our laboratory room shown in Figure 1 that will be the example environment of the experiments in this paper. In this figure, there are a number of hierarchical models of the room; the model at level 0 is the most abstract and the one at level 3 is the most concrete (so primitive). The plan at a certain level is generated based on the corresponding model. Initial states and goals are first described in the most abstract (or uppermost) model, and a number of task sequences are generated to achieve these goals in this model. (An example of the task hierarchy established in accordance with the model hierarchy is shown in Figure 2.) This plan generation is usually based on the descriptive information represented in the corresponding model. One of the sequences is selected according to a particular planning strategy (the utility is used in the case of DTP), and then each task in the sequence is further refined, that is, the sub-task sequences in the less-abstract model for achieving the task are generated. These sequences are called refinements of the task.

Actual conflicts are identified when all tasks have been expanded into primitive tasks, since the required amount of resources and time needed for executing the plan are precisely determined at this level. This may not prevent an agent from investigating the possibility of conflicts at an abstract level, however. For example, if a certain room is roughly modelled as a single object at an abstract level such as the level-0 model in Figure 1 and two agents have plans to work in this room at the same time, they can resolve this possible conflict by one agent deciding to work at another time. However, this conflict may not occur after the plans have been expanded into primitive ones, because it might turn out that the agents are able to work at different places in the room. In general, the process of conflict detection and resolution in abstract layers is simple because its domain model and related operators are simple. However, it usually results in redundant and inefficient plans.

Normal utilities for making efficient or high-quality plans do not usually take into account possible conflicts with other agents. As a result, although they can create acceptable plans when there is no interference between plans, they might not be able to do so when there is interference. Furthermore, in applications where real-time performance is stipulated, it is preferable that agents predict which conflicts will vanish or be easy or difficult to resolve during the remainder of the planning period. It is important, therefore, to provide another utility for plan selection when there is the possibility of conflict. However, determining what the conflicts are and which tasks easily cause them is a function of the location of scarce or heavily used resources and the type of agent; thus, the outcome strongly depends on the situation and environment where the sensor-network system is deployed. This type

[1] An agent selects the plan that may lead to the highest utility. However, the utility value is determined from the primitive task/plan, so the utility of a non-primitive task/plan is expressed as a range calculated according to the possible lower-level refined plans. It has been reported that agents should choose the plan that contains the highest utility and expand it to the next layer for effective planning (Goldwin & Simmons, 1998).

of information cannot be provided *a priori* during the design time. Therefore, agents have to learn an additional utility for MAS contexts.

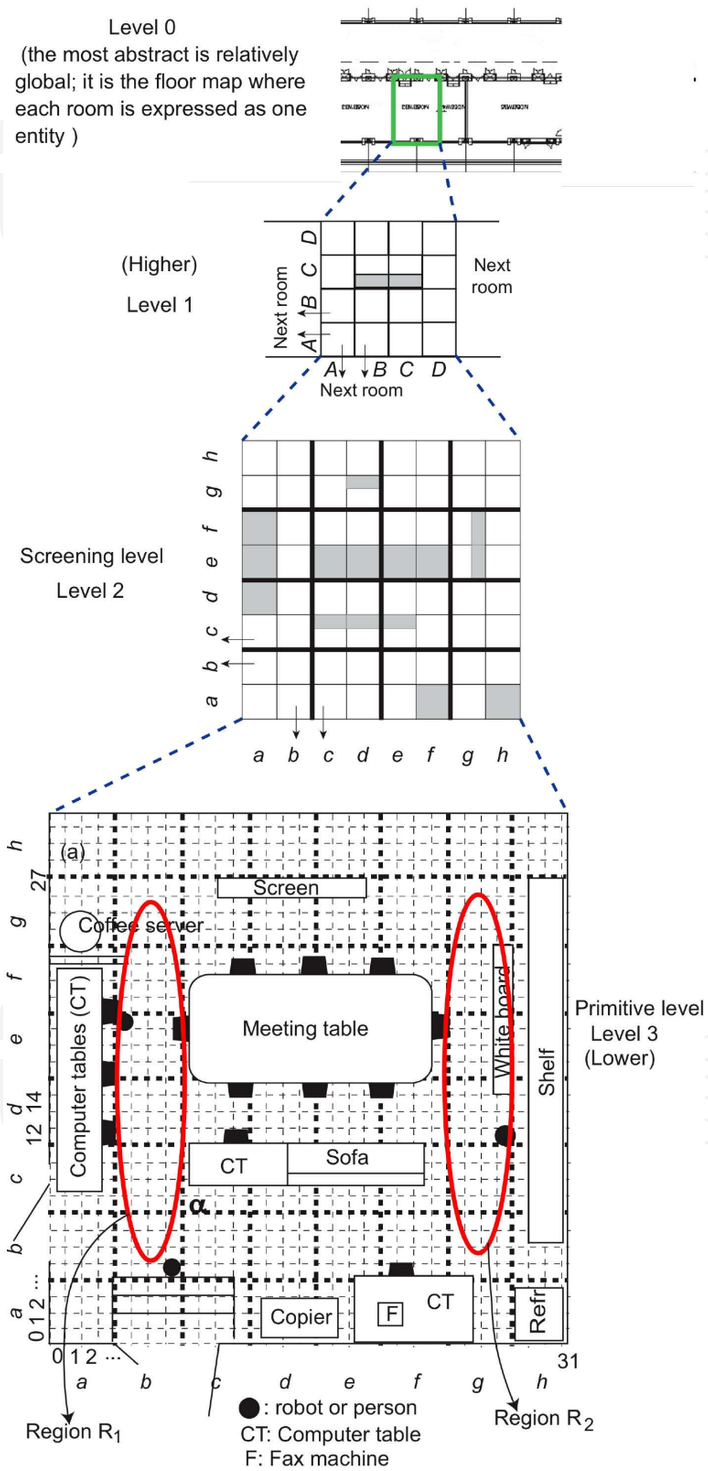


Fig. 1. Example of a hierarchical description.

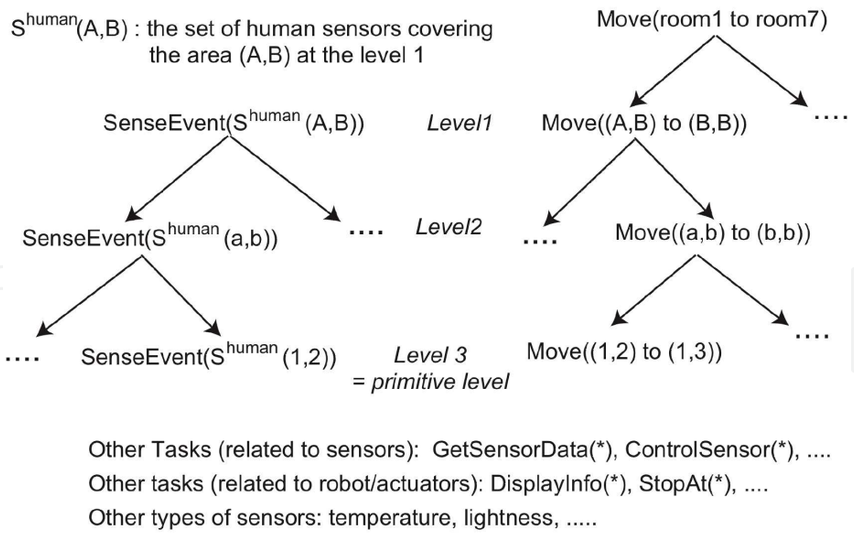


Fig. 2. Hierarchical task structure based on the hierarchical model.

3. Planning at the screening level

3.1 Planning architecture

In our planning architecture, agents first exchange only the presently being generated, scheduled and executed plans described in a certain abstract-level model, called the *screening level* (*SL*). We assume that the plans at this level are simpler than ones at the primitive level but are enough to classify the conflicting situations. The SL plans presently scheduled or being executed are called *SL-valid* plans, and the SL plans that are currently being generated (so are pending) are called *SL-pending* plans.

When agent a_i starts to create its plan for this environment, it first generates a number of SL plans (from the abstract-level plan) and tentatively selects one of them (using conventional utility). It then requests SL-valid and SL-pending plans from other agents to investigate the possible conflicts between a_i's new plan and other plans, by using an estimation based on the utility with the learned conflict discount as described in Section 4. According to this result, a_i selects one of the SL plans to refine further. This plan is marked as `SL-pending'. If $a_i$ is requested to send its plans during this process, it immediately notifies the request for that it is `SL planning' and sends the SL-pending plan right after it is determined. Agent a_i then waits for a short while for other unreceived plans; if it receives no other SL plans that have high conflict discounts, it proceeds to the next stage described below. Otherwise, one of the agents selects another SL plan instead of the current SL-pending plan; this may slow down the system in an extremely busy environment, so a tailored method for this issue will need to be developed in the future.

For further conflicts analysis, agent a_i requests primitive plans only from the agents whose plans are predicted to conflict with a_i 's SL plan. Then a_i modifies the primitive plan to eliminate the detected conflicts. If conflict discount is sufficiently learned, the cost of conflict resolution is relatively low and the resulting plan is acceptable. When a_i completes a primitive plan without conflicts, the plan is scheduled or executed immediately; and its SL-plan is marked `SL-valid'. Section 4 discusses how a_i learns to predict conflicts at the SL and how the utilities with a conflict discount are estimated.

We focus on applications where the same or similar plans are frequently reproduced. Examples of target applications are planning for the intelligent behaviours in sensor-network and ubiquitous-computing systems with many devices, such as sensors, effectors and robots (Figure 1), where agents reside in these devices to control them (Takada et al., 2003). Examples of application scenarios are described in (Kurihara et al., 2005). In this sort of application, e.g., robots moving in a room and assisting in people's daily activities, certain actions are repeated. We assume that other plans that are already scheduled or being executed are not modified (at least, the plans that have already been approved should be preferred) in the current implementation. Often this restricts the quality of the resulting plan. Our aim, however, is to select the most appropriate SL plan in a timely manner. If all of the plans generated at the SL appear to have high-discount conflicts, the agent can backtrack and select another plan at the SL or at a higher level; the agent still creates an abstract plan, which is simpler than creating a useless primitive plan, so we believe that the cost is not so high.

3.2 Conflict detection at screening level

The agent detects possible conflicts, according to resource and task information at the SL, by identifying the possibility of whether multiple plans will use the same resources, such as locations (e.g., squares in Figure 1). An example is illustrated in Figure 3, for which the SL is level 2 in Figure 1; a square at this SL (specified by a pair of lower-case letters) corresponds to 4x4 squares in the primitive model (A square in the primitive level is specified by a pair of positive integers.). In Figure 3, the agent can suggest that task t_l = *move*(***cd*** $\rightarrow$ ***dd***) in the new plan may conflict with task t'_n = *move*(***cd*** $\rightarrow$ ***bd***) in the SL-valid plan, where *move*(***cd*** $\rightarrow$ ***dd***) is the SL plan expressing the agent's movement from somewhere in area ***(c, d)*** to area ***(d, d)***. This conflict can be expected if some squares in area ***(c, d)*** can be simultaneously occupied by two agents during a certain time interval.

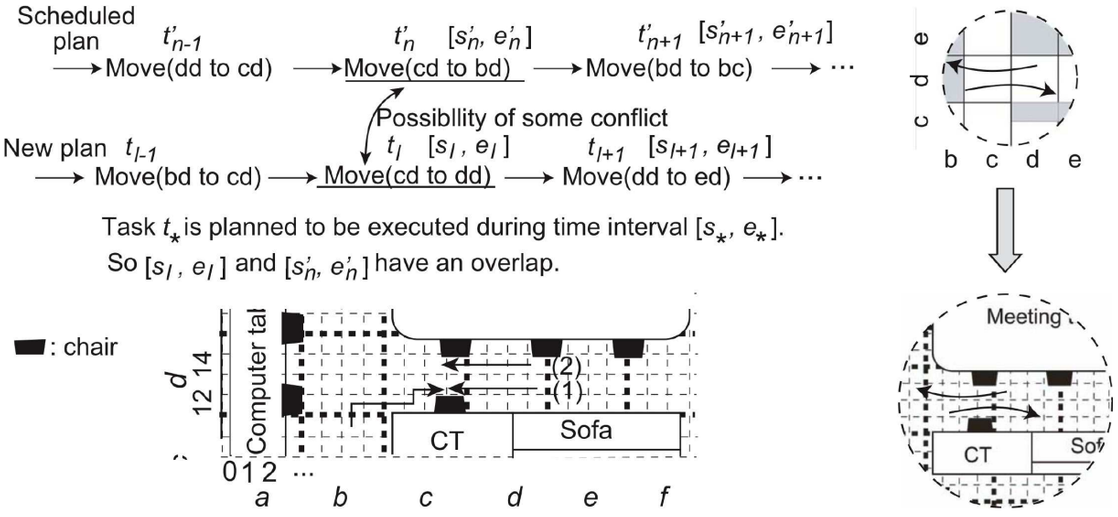


Fig. 3. Example of a detected conflict.

An agent has to take into account time relationships between tasks in the plans. The duration of each task in the SL-valid plans has already been determined, but not that of the

new plan. Thus, it uses the expected average duration of each SL task. This value is initially given as part of the SL model; for example, *move*(***cd*** $\rightarrow$ ***bd***) takes four ticks if agents (that is, robots) can move to the next small square in a tick. The expected duration is then statistically adjusted according to the generated primitive plans induced from this SL task.
The questions of when and where conflicts likely occur and whether their resolutions are difficult depend upon the system's environment. Suppose that three agents want to pass through area ***(b, d)***. In the SL model, this area (place is a resource) is expressed as a single entity, so conflicts can be expected. However, this area has enough room for three agents if each agent occupies a small square at the primitive level; hence, the conflicts might not actually occur or might be easily resolved. However, in ***(c, d)*** where agents move only left or right, there is not enough room for three agents. Thus, it seems probable that the agents' plans will have conflicts there. Of course, this probability is influenced by the temporal relationships of the agents entering area ***(c, d)***. If a conflict is detected, one of the agents must step out of the other agent's way and wait for it to pass by before resuming its movement.

Method	Description
Synchronization	Stop until another agent performing a task that requires a needed resource finishes the task and releases the resource. Wait for a primitive task or use of some resource by another agent until the task finishes or the agent releases the resource. This method may insert a number for "wait for a tick" for synchronization.
Waiting	Stop until other agents finish tasks that create pre-conditions of the local task. This method may insert a number for "wait for a tick" for synchronization.
Replacement	Replace tasks whose post-conditions do not affect tasks in other agents or whose pre-conditions are not affected by tasks in other agents. This method may replace the conflicting task with others, but these other tasks usually have lower utility (or incur extra cost).
Reordering	Reorder tasks to avoid negative relationships.
Insertion	Insert tasks whose post-conditions recover the pre-conditions of the task. This method adds some tasks, so the utility of the resulting plan decreases.
Commission	Entrust the task to other agents. This form of resolution is preferable when, for example, a conflict can only be resolved by other agents, or if another agent can do the task at lower cost. This method can eliminate some tasks, though some communications, not only for detecting the sharable tasks of the plans but also for committing them to another agent, take place.

Table 1. Examples of methods of resolution.

3.3 Conflict detection and resolution

A number of resolution methods, shown in Table 1, are applied to resolve conflicts. Thus, the agents involved must negotiate which agent (or all agents involved) should commit to modifying their plans and then decide what methods should be applied. These resolution

methods are defined as rules and applied under a certain policy. The resulting plans usually have extra cost for the resolutions. In this paper, we do not care what kind of policy is used; our only concern is the cost of resolution and the quality of the resulting plan.

4. Conflict estimation from conflict patterns

4.1 Conflict pattern --- an expression of conflicting situations

A *conflict pattern* (*CP*) expresses a conflict between SL plans. First, we focus on an SL task identified as having a conflict. Let t be an SL task in a new SL plan p, denoted by $t \in p$. Suppose that SL plans $p_1, \ldots, p_k$ of other agents are SL-valid. Then CP, denoted here by $\mathcal{P}(t)$, is expressed as

$$\mathcal{P}(t) = (t, (t'_1, o_1), \ldots, (t'_h, o_h))$$

where $t'_i \in p_j$ $(1 \leq \exists j \leq h)$ and o_i is optional data. CP describes the situation where t is expected to conflict with $t'_1, \ldots, t'_h$ in SL-valid plans.

The optional data o_i can be any information that can be used to distinguish conflicting situations more accurately. For instance, it may be information about (relative) the time of execution and agents' names or types that suggest their ability/performance or physical size (when agents, such as robots and vehicles, have physical bodies). In the example of Figure 3, CP is expressed as

$$\mathcal{P}_1(t_l) = (t_l, (t'_n, \ (\max(s'_n - s_l, 0), \min(e_l - s_l, e'_n - s_l)))),$$

where the optional data is the relative time interval during which the expected conflict may occur. To simplify the expression of this example, we describe the optional data in a more abstract form. For this purpose, we can use the expressions of time relativity; the duration of t'_n overlaps the anterior half (*ah*) or posterior half (*ph*) of the duration of t_l. Other cases of time relativity are expressed as "overlap (*ol*)." Thus, $\mathcal{P}_1(t_l) = (t_l, (t'_n, r'_l))$, where $r'_l = ah, ph$ or ol.

The situation in Figure 4 shows that t_l may conflict with t'_{n+1} and t''_{m-1}. The following CP corresponds to this situation:

$$\mathcal{P}_2(t_l) = (t_l, (t'_{n+1}, r'_l), (t''_{m-1}, r''_l))$$

where $r'_l, r''_l = ah, ph$ or ol.

4.2 Concept of conflict discount

Let $U(p)$ (or $U(t)$) be the initial utility for a primitive plan p (or a primitive task t). $U(p)$ for a non-primitive plan (or task) is the range that cumulatively indicates possible lower-primitive plans/tasks. We introduce the *conflict discount* for a CP, $cd(\mathcal{P})$. The conflict discount is conceptually defined as

$$cd(\mathcal{P}) = U(pp) - U(pp_m) + CCR(\mathcal{P}) \quad (1)$$

where pp is the primitive plan of SL plan p before conflict resolution, and pp_m is the modified primitive plan for resolving conflict $\mathcal{P}$. The term CCR indicates the cost of conflict detection and resolution at the primitive level, which is calculated by combining the costs of requesting, receiving, and analyzing primitive plans from other agents and applying conflict resolution rules to modify the new plan. So even if no conflict actually occurs at the primitive level ($U(pp)=U(pp_m)$), $cd(\mathcal{P}) \neq 0$. This is because, if a conflict is expected at SL, the cost of conflict detection will be incurred. Define $cd'(\mathcal{P}) = U(pp) - U(pp_m)$ as the difference in utilities. The estimation of $cd(\mathcal{P})$ is described in the next section.
When an agent has a new SL plan p that is expected to have CPs, $\mathcal{P}_1$, …, $\mathcal{P}_N$,

$$cd(p) = \sum_{i=1}^{N} cd(\mathcal{P}_i).$$

The agent uses the modified utility $U(p) - cd(p)$ instead of $U(p)$. When no conflicts are predicted, the agent uses $U(p)$ since $cd(p) = 0$. Our method statistically adjusts the conflict discounts for frequently appearing CPs. Because we focus on the efficiency of plans, we assume that $U(p)$ is the estimated execution time of the primitive plan in the example below.

4.3 Estimation of conflict discount

The conflict discount for a CP, $cd(\mathcal{P})$, is iteratively adjusted by the average or update function as follows when CP is observed s times.

$$cd_s(\mathcal{P}) = \sum_{i=1}^{s} \frac{d_i}{s} \tag{2}$$

$$cd_s(\mathcal{P}) = \lambda * cd_{s-1}(\mathcal{P}) + (1 - \lambda) * d_s \tag{3}$$

where $0< \lambda < 1$ and d_s indicates the s-th CCR_s plus the s-th observed utility difference between the original primitive plan and the plan after the resolution of the conflict corresponding to $\mathcal{P}$. Eq. (3) is more sensitive to environmental changes than Eq. (2). Note that the conflict of $\mathcal{P}$ might not occur at the primitive level after all; if so, $d_s = 0 + CCR_s$. For example, if the partner agent takes route (1) in Figure 3, and this conflict can be resolved by taking a detour or by using "wait for two ticks" to wait until the partner agent passes by. In this case, $d_s = 2 + CCR_s$. However, if the partner agent takes route (2) in Figure 3, no conflict actually occurs and $d_s = 0 + CCR_s$.
To acquire the CCR value for each plan, we assume that agents can monitor their planning activities by themselves. More precisely, CCR consists of the time for (1) requesting and receiving primitive plans from other agents that are suggested to have conflicts, (2) detecting actual conflicts between these plans and the local plan, and (3) modifying the local plan to resolve these conflicts. Agents keep the times for these activities. The conflict discount is re-calculated using the value of CCR plus the differential utility for each CP acquired by each agent from Eq. (2) or (3).

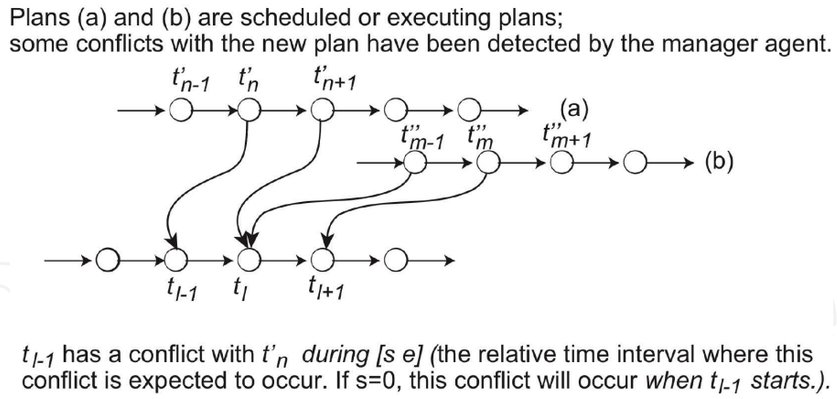


Fig. 4. Example of conflicts between plans.

The calculation of $cd(p)$ of SL plan p, like the conflict resolution process, is an iteration of the procedures for (1) searching for, from the first task, the task t that has a conflict pattern $\mathcal{P}$ with other plans, and (2) predicting the conflict discount $cd(\mathcal{P})$. In procedure (2), the additional cost of avoiding conflicts is predicted, and thus the start times of subsequent tasks may be delayed for this amount of time. Since a number of conflicts may appear and disappear in the remaining part of the plan because of this delay, the agent detects the next conflicting task by using the adjusted duration.

4.4 Sub-conflict patterns

It is probable that many CPs will be created, and storing many CPs in the casebase would require a large amount of memory. This also incurs a large search cost, which degrades scalability. It also lowers the performance of conflict estimations of the CPs. Here, we can try to reduce the memory taken up by the CPs.

Suppose that $\mathcal{P}_1$ and $\mathcal{P}_2$ are CPs:

$$\mathcal{P}_1 = (t, (t_1, r_1), \ldots, (t_n, r_n))$$
$$\mathcal{P}_2 = (t', (t'_1, r'_1), \ldots, (t'_m, r'_m))$$

If $t = t'$ and $\{(t_1, r_1), \ldots, (t_n, r_n)\} \subset \{(t'_1, r'_1), \ldots, (t'_m, r'_m)\}$, then $\mathcal{P}_1$ is the sub-conflict pattern (sub-CP) of $\mathcal{P}_2$, denoted by $\mathcal{P}_1 \subset \mathcal{P}_2$. Now, we assume that $cd(\mathcal{P}_1) \leq cd(\mathcal{P}_2)$ if $\mathcal{P}_1 \subset \mathcal{P}_2$. This is a natural assumption because $\mathcal{P}_1$ is resolved if the conflict with $\mathcal{P}_2$ is resolved.

To save memory, the agent only stores CPs whose conflict discount values are near the turning point of the decision. For example, if $cd(\mathcal{P}_2)$ is sufficiently small, the cd value for $\mathcal{P}_1$ ($\subseteq \mathcal{P}_2$) will not necessarily be stored, so its cd estimation can be eliminated. Similarly, if $cd(\mathcal{P}_1)$ is large, which means that the agent will give up the current SL plan, the cd value for $\mathcal{P}_2$ ($\supseteq \mathcal{P}_1$) does not have to be stored.

5. Experiments

5.1 Conflict discount estimation

We experimentally investigated how cd' (instead of cd) changes depending on the ways that agents interfere in a simulated laboratory room (Figure 1). Agent A randomly selects a

starting point in region R_1 and a goal in region R_2 and then tries to generate a new plan for this movement. Another agent, B, already has an approved plan whose start and goal are also randomly selected in R_1 and R_2. In this setting, these agents do not cause any conflict when they may take different routes, such as to the north or south of the meeting table. However, they are likely to have conflicts when they both have to pass through area ***(c, d)*** because chairs and computer tables slightly narrow the route through it. Hence, we focused on the cases in which a conflict would be expected there at the SL and iterated the experiment until A's task *move*(***cd*** $\rightarrow$ ***dd***) conflicted with B's task, which were both expressed as *move*(***cd*** $\rightarrow$ ***dd***) (same direction) at the SL. Note that the duration of A's SL plan was an estimated value that may differ from the actual duration of execution. This estimated duration was not used in the experiments in (Sugawara et al., 2005); thus, some of the experimental values shown below are slightly different from the ones reported in that paper. The SL plan was expanded into a primitive plan, and we investigated the conflict discount after conflict resolution. Because B requests A's primitive plan, extra costs may be incurred even if no conflicts end up occurring. Therefore, in the following experiments, the number of plans of other agents that were predicted to have conflicts with the new plan was used as the approximate value of CCR (hence, a constant for each $\mathcal{P}$) of Eq. (1), because it is proportional to the number of these plans. This assumption means that it takes a tick to request and receive a primitive plan from another agent, check for conflicts between the received and local plans, and resolve these conflicts. We iterated this experiment a few hundred times to calculate cd'.

The task *move*(***cd*** $\rightarrow$ ***dd***) usually takes four to six ticks in this environment. Note that we assumed the SL-task *move*(X,Y) during interval [s, e] occupies resource X during s to e and resource Y at e and that the primitive-level task *move*(x, y) during [s, s+1] (a primitive task takes 1 tick) occupies x and y during s to s+1. If the agent finds a possible conflict within the first two ticks, the relative time relationship is denoted by *ah*; Additionally note that if it finds such a possibility within the last two ticks, the relative time relationship is denoted by *ph*. Otherwise, the relative time relationship is denoted by *ol*. Hence, we estimated the values of cd' for the following conflict patterns:

$$\mathcal{P}_3 = (move(\boldsymbol{cd} \rightarrow \boldsymbol{dd}), ((move(\boldsymbol{cd} \rightarrow \boldsymbol{dd}), r))),$$

where r is *ah*, *ph* or *ol*, meaning that these two agents move in the same direction.

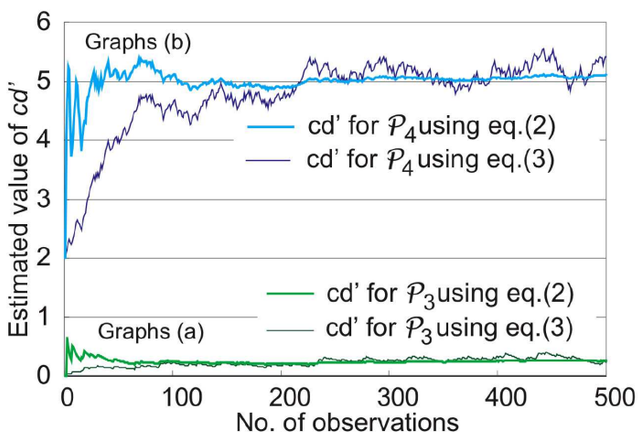


Fig. 5. Estimated cd' and average values.

Tables 1 and 2 show the average values from ten experiments based on ten different random seeds, and the graphs in Figure 5 are from one of these experiments.
Graphs (a) in Figure 5 show the estimated values of cd'_m (λ = 0.98, $1 \leq m \leq 500$) derived from Eqs. (2) and (3) when $r = ol$. In these cases, $cd'(\mathcal{P}_3) = 0.71$ (so $cd(\mathcal{P}_3) = cd'(\mathcal{P}_3) + CCR(\mathcal{P}_3) = 1.71$), which is reasonably small. This is because the two-square-wide path is wide enough for two agents to pass through the area, but agent *A* sometimes has to take a detour to avoid conflicts. Other cases, such as moving in the opposite direction, are shown in (Sugawara et al., 2005).
However, the *cd'* values largely differ when two agents, *B* and *C*, which have approved plans (i.e., plans that do not conflict with each other), move in the same direction $move(\boldsymbol{cd} \rightarrow \boldsymbol{dd})$ and agent *A* begins to create a plan to move in the opposite direction though the same area. The conflict pattern of this situation is expressed as

$$\mathcal{P}_4 = (move(\boldsymbol{cd} \rightarrow \boldsymbol{bd})), ((move(\boldsymbol{cd} \rightarrow \boldsymbol{dd})), ol), (move(\boldsymbol{cd} \rightarrow \boldsymbol{dd})), ol))).$$

The estimated $cd'(\mathcal{P}_4) = 5.12$ ($cd(\mathcal{P}_4) = 7.12$) is quite different from the previous cases, as shown by graphs (b) in Figure 5. Because *B* and *C* move almost simultaneously without conflicts, they usually occupy the narrow route in area ***(c, d)*** together. Thus, agent *A* always has to move aside, wait for several ticks until *B* and *C* pass, and then move back to the original route. If the agent's new plan is predicted to have this conflict pattern at the SL, it can select, after learning, another route, such as one taking it north of the meeting table or another taking it south of the sofa in Figure 1, provided the route is shorter than the one in the original plan plus 7.12.
Table 2 shows the estimated *cd'* values in time-relativity cases other than $\mathcal{P}_4$. For example, if one of the relative time relationships in $\mathcal{P}_4$ is *ah* (This CP is denoted by $\mathcal{P}'_4$), the estimated $cd'(\mathcal{P}'_4) = 1.80$. This is small because if *B* and *C* move a slight distance away from each other, *A* can weave its way around them. In the case of *ph-ph*, *A*'s planned task $move(\boldsymbol{cd} \rightarrow \boldsymbol{bd})$ may conflict in the latter half of its execution, so the agents will usually not meet in the narrow area (*A* moves right to left). However, because of uncertainty, they infrequently meet at different times in the narrow area. Table 2 suggests that the values of *cd'* depend on the resource structure of the routes, especially area ***(c, d)*** in Figure 1.

	ol-ol ($\mathcal{P}_4$)	*ph-ph*	*ah-ah*	*ah-ol* ($\mathcal{P}'_4$)	*ah-ph*	*ph-ol*
Value of *cd'*	5.12	3.67	3.30	1.80	0.75	1.87

Table 2. Experimentally estimated conflict discount *cd'*.

Suppose that in another situation the agent finds a CP, $\mathcal{P}_5$ such that $\mathcal{P}_4 \subset \mathcal{P}_5$. This CP may appear when conflict among more than four agents at ***(c, d)*** is expected. In this case, $cd'(\mathcal{P}_5)$ must be larger than 5.12. If this value is larger than the predefined threshold, the agent can calculate that $cd(\mathcal{P}_5) \geq 7.12$ (or $cd(\mathcal{P}_5) \geq 8.12$ if this conflict occurs among more than four agents), suggesting that it should try to find another route or shift (delay) its start time to avoid this conflict, even if it has no data about $\mathcal{P}_5$. Conversely, $cd'(\mathcal{P}'_4) = 1.80$ can induce $cd'(\mathcal{P}_3) \leq 1.80$. If this value is small enough, the agent does not need to calculate $cd(\mathcal{P}_3)$. Table 2 also indicates that $cd'(\mathcal{P}_3) \leq 0.70$ if $r = ah$ or ph in $\mathcal{P}_3$.

5.2 Cost (length) of generated plans

We investigated how efficient plans are generated with lower cost after a conflict pattern is found. In our planning strategy, agent *A* tries to select or generate another SL plan that is expected to have no conflict with other plans and whose estimated utility (in our case, the length of the plan) is less than the estimated utility of the original SL plan plus *cd* (if the CP is $\mathcal{P}_4$, then $cd(\mathcal{P}_4)$ is 7.12). The cost of selecting or generating another SL-plan is relatively low because we can set the upper limit of plan length. If *A* can find the new SL plan, it is selected and further refined. If *A* cannot find one, the original plan is selected (so conflict detection and resolution may be required). In the conventional planning strategy, the first SL plan to be generated would always be refined even if some conflicts were expected. (Of course, there might be no conflicts after all).

We examined, in our simulated room, the improvement of our planning strategy that resulted from using the estimated conflict discount value in Table 2. The results of this experiment (Table 3) show that our planning strategy provides an improvement of 2.65 ticks on average when a conflicting situation corresponding $\mathcal{P}_4$ is detected. In other cases, our planning method can generate efficient plans except when the conflict time relativity is *ph-ol*. This improvement is not very large. However, the ability to provide some information for deciding whether the agent should continue to refine the current plan even if the conflict resolution process will very likely be invoked or try to find another plan that does not have conflict with other agents is significant in applications like ours. In the *ph-ol* case, *cd'* is low so *A* cannot find any other better route.

CPs	Conventional strategy	Our planning strategy	Improvement
$\mathcal{P}_4$ (*ol-ol*)	33.34	30.69	2.65 %
ah-ah	32.39	30.44	1.95 %
ph-ph	30.93	29.40	1.53 %
ph-ol	23.80	23.80	0 %

Table 3. Cost (length) of resulting primitive plans. Columns 1 and 2 respectively show the average cost of primitive plans derived from the original SL plans and that of primitive plans derived under our planning strategy. In both cases, the cost of conflict detection and resolution is included.

The improvement shown in Table 3 seems fairly small, but our simulated laboratory room is based on an actual room; we believe that our method would be more significant in other situations/environments. For example, (1) if more robots were to move right to left in the narrow area in Figure 3, (2) if the chair there were a bench (a longer chair), or (3) if there were a shorter detour, the improvement would be larger, thus the resulting plans would be of relatively higher quality than the ones obtained by a conventional planning strategy. We finally note that, although the start and goal positions were selected randomly in our experiments, agents (including persons) in actual applications usually have fixed start and goal points. Therefore, we believe that the improvements derived from the experimental results would appear more when this is actually applied to this kind of systems.

6. Discussion and related work

There have been a number of studies on efficient planning in the MAS context. For example, GPGP (Decker & Lesser, 1992) is a general framework for generating effective plans using task and resource relationships among agents. Our method can be used in this framework to identify which abstract plan (task) should be refined first so that the map of the task relationships related to the plan can be created.

Hierarchical planning and coordination issues for improving MAS planning have also been discussed. For example, Ref. (Clement et al., 2001) proposed choosing the most appropriate abstract task/plan on the basis of summary information derived from the primitive tasks and plans in a bottom-up fashion. This method can avoid hopeless planning if some resources are recognized to be insufficient at an abstract level. It also introduced *fewest-threats-first* (FTF) heuristics to choose a lower (deeper) plan. Our approach focuses on the cases where conflicts can be accurately identified at only deeper levels, because the tasks, resources, and their environment in an abstract model are described in an abstract way. Furthermore, a plan with fewer conflicts does not always lead to a better plan; it is possible that only one conflict fails to be resolved but that conflict is nonetheless a critical one. The idea behind our research is that, although conflicts may be invisible at abstract levels (including the SL), there is a tendency that conflicts often occur depending on the environmental factors related to the availability and use of resources, such as the location of agents, the kind of resources, and type of agents, as well as on the kind of task. Hence, we aim at expressing and distinguishing these situations by using CPs in order to enable agents to statistically learn the difficulty of conflict resolution and the quality of a resulting plan.

A number of issues related to MAS planning have been investigated in case-based reasoning (CBR) or its related domains. For example, (Giampapa & Sycara, 2001) proposed a conversational case-based reasoner, called NaCoDAE, which is a type of agent in their MAS applications and helps users decide a course of action by engaging them in a dialogue in which they must describe the problem or situation of assigning missions to platoons. Plan reuse for the same/similar situations in a MAS context has also been proposed for MAS coordination (Sugawara, 1995) and collaboration (Plaza, 2005). A remarkable work similar to our approach is (Macedo & Cardoso, 2004), where a case is used to expand an abstract plan to a less abstract one in HTN, although we focus on avoiding conflicts and/or selecting costless conflicts. In this sense, our motivation is more similar to that in (Aha et al., 2005) which applied CBR to a real-time strategy game.

Our work is also related to hierarchical reinforcement learning, such as (Dietterich, 1998; Kaelbling, 1993; Sutton et al., 1998), because an abstract task is considered to be a subroutine or a subfunction to be learned. For example, in the MAXQ approach (Dietterich, 1998), a task is divided into subroutines that are individually learned by RL methods. Our approach is to select an appropriate subroutine for each situation. In MAXQ, the conflict discount is assumed to have been learned at lower levels. However, in a multi-agent setting, it is naturally difficult to define the task hierarchy for all agents simultaneously.

One clear limitation of our method is that the reliability of *cd* values heavily depends on the accuracy of the SL conflict detection and time-estimation processes. Thus, it is very important to select the appropriate SL and carefully describe the SL model. For example, if level 1 in Figure 1 is the SL, our method does not work well since that level is too abstract. As mentioned above, another issue is that the use of optional data in CPs is important for distinguishing one situation from another. To distinguish situations, our method needs the

location of task execution (which may determine available resources), type of agent (which may determine required resources), and (relative) time information. Additionally, if many CPs are expected in a plan, conflict detection at the SL may be ambiguous regarding the scheduled time and resources of the SL tasks, which would affect the quality and cost of the plans. Finally, our method will have to be extended before it can deal with situations where multiple plans are created simultaneously; this extension is important for effective planning, and it will be addressed in a future work.

7. Conclusion

This chapter proposed a method to predict, at an abstract level called the screening level, the cost of possible conflict resolution, and the quality of the resulting plan, to generate better primitive (concrete) plans. In our framework, an agent called the manager agent maintains the plans that are scheduled or being executed at the screening level and predicts possible conflicts between these plans and the newly proposed plan. Then, if necessary, a detailed analysis of primitive plans is performed by individual agents. We conducted experiments to reveal the estimated additional cost (estimated *cd* and *cd'* values) of the plans after conflict resolution and the efficiency of plans derived from our method. Our method enables agents to decide whether the current plan should be refined or another plan should be created at an earlier stage, that is, before an agent creates its primitive plan; this decision makes agents' planning efficient.

Acknowledgement: This research was supported by SCOPE program of the Ministry of Internal Affairs and Communications, Japan, under contract 071607001.

8. References

Aha, D. W.; Molineaux, M. & Ponsen, M. (2005). Learning to win: Case-based plan selection in a real-time strategy game, *Proc. of the Sixth International Conference on Case-Based Reasoning (ICCBR 2005)*, LNAI 3620, pp. 5 – 20.

Clement, B. J.; Barrett, A. C.; Rabideau, G. R. & Durfee. E. H. (2001). Using abstraction in planning and scheduling, *Proc. of 6th European Conference on Planning*.

Decker, K. & Lesser, V. (1992). Generalizing the Partial Global Planning Algorithm, *International Journal on Intelligent Cooperative Information Systems*, Vol. 1, No. 2, pp. 319 – 346.

Dietterich, T. G. (1998). The MAXQ Method for Hierarchical Reinforcement Learning, *Proceedings of the International Conference on Machine Learning (ICML 98)*, pp. 118 – 126.

Giampapa, J. A. & Sycara, K. (2001). Conversational case-based planning for agent team coordination, *Proc. of the Fourth International Conference on Case-Based Reasoning (ICCBR 2001)*, LNAI 2080, pp. 189 – 203.

Goldwin, R. & Simmons, R. (1998). Search Control of Plan Generation in Decision-Theoretic Planners, *Proc. of AIPS 1998*, pp. 94 – 101.

Erol, J. H. K. & Nau, D. S. (1994). HTN planning: Complexity and expressivity, *Proc. of the National Conference on Artificial Intelligence (AAAI 94)*, pp. 1123 – 1128.

Kaelbling, L. P. (1993). Hierarchical Learning in Stochastic Domains: Preliminary Results, *Proceedings of the International Conference on Machine Learning (ICML-93)*, pp. 167 – 173.

Kurihara, S.; Aoyagi, S.; Takada, T.; Hirotsu, T. & Sugawara, T. (2005). Agent-Based Human-Environment Interaction Framework for Ubiquitous Environment, *Proc. of the International Workshop on Networked Sensing Systems*, pp. 103 – 108.

Kurihara, S. (2008). Human Behavior Mining using Sensing Network, *Proceedings of the First International Workshop on Content Creation Activity Support by Networked Sensing (CCASNS08)*.

Macedo, L. & Cardoso, A. (2004). Case-Based, Decision-Theoretic, HTN Planning, *Proc. of ECCBR 2004*, LNAI 3155, pp. 257 – 271. Springer-Verlag.

Plaza, E. (2005). Cooperative reuse for compositional cases in multi-agent systems, *Proc. of the Sixth International Conference on Case-Based Reasoning (ICCBR 2005)*, LNAI 3620, pp. 382 – 396.

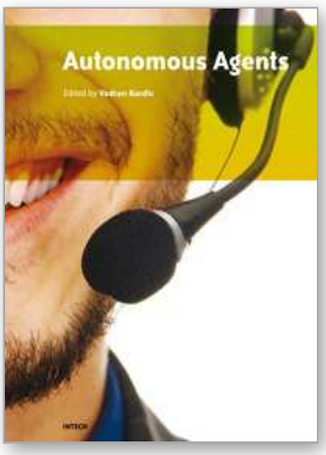
Sacerdoti, E. (1974). Planning in a hierarchy of abstraction spaces, *Artificial Intelligence*, Vol. 5, No. 2, pp.115 – 135.

Sugawara, T. (1995). Reusing Past Plans in Distributed Planning, *Proc. of the 1st International Conference on Multi-Agent Systems* (*ICMAS95*), pp. 360 – 367.

Sugawara, T.; Kurihara, S.; Hirotsu, T.; Fukuda, K. & Takada, T. (2005). Predicting Possible Conflicts in Hierarchical planning for Multi-Agent Systems, *Proc. of 4th International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS 2005)*, pp. 813 – 820.

Sutton, R. S.; Precup, D. & Singh. S. (1998). Intra-Option Learning about Temporary Abstract Actions, *Proceedings of the International Conference on Machine Learning* (*ICML98*), pp. 556 – 564.

Takada, T.; Kurihara, S.; Hirotsu, T. & Sugawara, T. (2003). Proximity Mining: Finding Proximity using Sensor Data History, *Proc. of IEEE Workshop on Mobile Computing Systems and Applications*, pp. 129 – 138.



Autonomous Agents
Edited by Vedran Kordic

ISBN 978-953-307-089-6
Hard cover, 130 pages
Publisher InTech
Published online 01, June, 2010
Published in print edition June, 2010

Multi agent systems involve a team of agents working together socially to accomplish a task. An agent can be social in many ways. One is when an agent helps others in solving complex problems. The field of multi agent systems investigates the process underlying distributed problem solving and designs some protocols and mechanisms involved in this process. This book presents a combination of different research issues which are pursued by researchers in the domain of multi agent systems.

How to reference
In order to correctly reference this scholarly work, feel free to copy and paste the following:

Toshiharu Sugawara, Satoshi Kurihara, Toshio Hirotsu, Kensuke Fukuda and Toshihiro Takada (2010). Effective Planning for Conflicting Situations for Ubiquitous Sensor Network Environments, Autonomous Agents, Vedran Kordic (Ed.), ISBN: 978-953-307-089-6, InTech, Available from: http://www.intechopen.com/books/autonomous-agents/effective-planning-for-conflicting-situations-for-ubiquitous-sensor-network-environments

INTECH
open science | open minds

InTech Europe
University Campus STeP Ri
Slavka Krautzeka 83/A
51000 Rijeka, Croatia
Phone: +385 (51) 770 447
Fax: +385 (51) 686 166
www.intechopen.com

InTech China
Unit 405, Office Block, Hotel Equatorial Shanghai
No.65, Yan An Road (West), Shanghai, 200040, China
中国上海市延安西路65号上海国际贵都大饭店办公楼405单元
Phone: +86-21-62489820
Fax: +86-21-62489821

IntechOpen