Multiagent and Grid Systems, Special Issue on Negotiation Mechanisms for Multiagent and Grid Resource Management Systems, Vol.4, No.1, 2008, 25-44
(Submitted 24 September 2006; Revised 10 December 2006)

Retractable Contract Network for Empowerment in Workforce Scheduling

E.P.K. Tsang[1], T. Gosling[1], B. Virginas[2], C. Voudouris[2], G. Owusu[2] and W. Liu[1]
[1]Department of Computer Science, University of Essex
[2]BT Research Laboratories

Abstract

This paper is about business modelling and negotiation protocol design in distributed scheduling, where individual agents have individual (potentially conflicting) interests. It is motivated by BT's workforce scheduling problem, in which multiple service providers have to serve multiple service buyers. The service providers and buyers all attempt to maximize their own utility. The overall problem is a multi-objective optimization problem; for example, one has to maximize completion rates and service quality and minimize travelling distances. Although the work is motivated by BT's business operations, the aim is to develop a general negotiation protocol for staff empowerment.

Standard contract net is a practical strategy in distributed scheduling where agents may have conflicting objectives. In this paper, we have introduced a retractable contract net protocol, RECONNET, which supports hill-climbing in the space of schedules. It is built upon a job-release and compensation mechanism. A system based on RECONNET has been implemented for BT's workforce scheduling problem. The software, which we call ASMCR, allows the management to exert full control over the company's multiple objectives. The manager generates a Pareto set of solutions by defining, for each buyer and seller, the weights given to each objective. ASMCR gives service buyers and sellers ownership of their problem and freedom to maximize their performance under the criteria defined by the management. ASMCR was tested on real-sized problems and demonstrated to meet BT's operational time requirement. It has full potential to be further developed for tackling BT's workforce scheduling problem.

Keywords: management, negotiation protocols, computational mechanism design, modelling, multi-objective optimization

1 Introduction

Empowerment in management means giving employees opportunities to exercise personal choices (Wilkinson 1998). Thomas and Velthouse (1990) suggest that through employee empowerment, the workforce could be better motivated. This paper studies staff empowerment in workforce scheduling. The task is to model the operation and design a negotiation protocol to enable the employees to generate schedules which are consistent with the management's objectives. Though motivated by a real-life application, the negotiation protocol proposed is designed to be general.

The proposed negotiation protocol is deigned for distributed scheduling. In many real life scheduling problems, multiple entities have to work together. Human factors may lead to local considerations not being conveyed to the central scheduler. For example, different entities often have conflicting interest. Should scheduling be done centrally, independent entities do not always inform the central scheduler of all the relevant information (game playing). Besides, allocation of jobs may be based on unwritten rules. For example, a service seller manager (to be explained below) may take staff emotion into consideration and apply discretion in scheduling his staff. In such situations, centralised scheduling is very difficult. Partial information could lead to sub-optimal schedules being generated. Staff involvement in job allocation could lead to higher morale and higher efficiency. This motivates BT to look at distributed workforce scheduling seriously.

From an optimization point of view, the business operation involves many optimization problems. Each entity looks after its only interest. Distributed constraint satisfaction is a well studied topic which is highly relevant to distributed scheduling. Most of the work in distributed constraint satisfaction involves cooperative agents, where agents work together to achieve some common goals; for example see Durfee (1999), Prosser (1990), Yokoo and Ishida (1999). In this paper, BT's problem is formulated as a distributed scheduling problem (see Ursu, Virginas and Voudouris (2004), Virginas, et al. (2005)). It is worth pointing out that the task of extracting knowledge from BT's operation and formalising the problem is nontrivial. This requires expertise in the operation as well the management's philosophy. After formulation of the problem, the task is to find a negotiation protocol for achieving the management's goals. This work is related to research in bargaining (e.g. see Sim and Choi (2003), Sim (2005), Ito et al (2005)), mechanism design (e.g. see Sandholm 2003) and distributed constraint satisfaction (e.g. see Faltings (2005), where agents may have conflicting goals).

BT's workforce is organized in regions. Each region has a Service Buyer (or *Job Agent*) and a Service Seller (or *Engineer Agent*). These agents are managed by a central manager. Each service buyer has a set of jobs to be completed. Each service seller manages a set of engineers, who can be assigned to jobs. The service buyers buy services from the sellers by inviting them to bid for individual jobs. The service sellers base their bidding on which engineers could service which job (depending on availability and skills), the engineers' preferences and their travelling distances to the jobs. The buyer selects bids based on preferences and travelling distances (information supplied by the sellers). Offers are made to selected bids. These offers are binding, which means the buyer will have to honour a contract as soon as it is accepted by the seller. The sellers may receive multiple offers that require the same engineer. In this case, it uses its utility criteria to select offers. The question is how to design a protocol to enable efficient assignments of engineers to jobs.

The buyers and sellers must maximize their utility. The manager is responsible for maximizing the company's utility. The manager is able to achieve this by defining the utility function of each buyer and seller. The manager's problem is a multi-objective optimization problem: it has to strike a balance between job completion rates, service quality, travelling distances and other objectives, which will be elaborated later.

2 Modelling BT's workforce scheduling problem

2.1 Overview of the problem

The first step towards tackling BT's workforce scheduling problem was to model it. This is one of the most laborious parts of the project so far. Understanding of the business operations is crucial to modelling. The current operation is centralized. Staff empowerment motivates BT's research into distributed scheduling: the aim is to raise productivity through raising morale through empowerment.

Modelling is arguably the most important part of problem solving, as it determines what techniques one can and cannot apply to a problem (e.g. see Freuder 1999, Borrett and Tsang 2001). The following is a specification of the components in BT's workforce scheduling problem. The challenges are also highlighted:

1. **The Manager**

   As mentioned above, the manager's problem is a multi-objective optimization problem.

Our aim is to provide the end-user (the human manager who is in charge of day-to-day running of BT's workforce scheduling) with a decision support tool. For convenience, we call this tool "the manager". The role of the manager is to build a Pareto set of schedules for the end-user to choose from.

The quality of a Pareto set can be defined in many ways (see, for example, (Okabe 2004), for a good survey). In this project, the criteria for measuring the quality of Pareto sets are range and coverage. Range refers to the inclusion in the Pareto set solutions that optimize each of the individual objectives. Coverage refers to even distribution of solutions. For example, in a 2-objectives maximization problem, the Pareto set {(10, 10), (20, 20), (30, 30)} has a better range than, say, {(10, 10), (11, 11), (12, 12)}, and a better coverage than, say, {(10, 10), (11, 11), (30, 30)}.

The manager attempts to find a "good" Pareto set by the following means:

- Let P be the Pareto set of schedules found so far. P is initialized to an empty set.
- Repeat for a predetermined amount of time:
- Based on the Pareto set of schedules found so far, the manager determines the weights of each objective for each buyer and seller. Each buyer and seller may be given different amalgamation functions.
- The manager passes to each buyer and seller the weights of each objective.
- Through negotiation, the buyers and sellers will arrive at a schedule, which will be returned to the manger and added to P.

2. Service Buyers

Each Buyer handles a set of jobs that need to be done. It has to find a contract for each job, by communicating with service sellers. The Buyer's job is to formulate invitations to Service Sellers to bids for jobs. The buyer may (a) invite bids for one job at a time, or (b) invite bids for more than one job at a time. In the latter case, jobs can be bundled[1]. When bids are received, the Buyer is responsible for allocating jobs to the appropriate Seller. The Buyer then makes offers for those bids (binding) but these might not be accepted by the Seller.

If needed, it can also contact other buyers for *contract-release*, which means offering

[1] For any job that takes more than x days to finish and is more than y miles from the engineer, accommodation allowance is given. It is possible that Sellers may want to bundle jobs that are located in close proximity geographically.

other buyers as well as the seller compensation to give up their contracts (this will be formally defined later).

The buyer has multi-objectives, but at any point, it is given an amalgamation function by the manager. So it is effectively dealing with a single objective optimization problem. A buyer would attempt to find good solutions to its problem through a local search, which is facilitated by the contract-release protocol.

The buyer's challenge is in how to hill-climb to improve its utility according to a malgamation function given by the manager.

3. Service Sellers

Each seller handles a number of engineers. Engineers may have different abilities and constraints. The Seller's job is to bid for jobs. A bid can logically be described as a pair (*Pref*, *Dist*), where *Pref* is the preference and *Dist* is the distance. It may return just one, or the set of all available bids, or a subset (possibly Pareto set) of (non-dominating) bids for each job. Note that there are typically more jobs than those that the engineers can handle. Sellers can offer partial solutions to requests, e.g. "we cannot do all 3 bundled jobs in those 5 days, but we can do the first 2 jobs in the first 3 days". In this paper, we assume that the sellers are relatively passive: when a seller receives invitations to bid for jobs, it is required to make all the non-dominated bids (as in multi-objective optimization). Then it waits for (binding) offers from the buyers, to which it may accept (binding) or decline. Each seller has multiple objectives, but, like the buyers, at any time, it is given an amalgamation function by the manager on the weights of each objective. So effectively, a seller deals with a single objective optimization problem. However, the seller does not know the buyer's weights on its objective, hence the return of all potentially non-dominated bids to the buyer. For example, the seller might return a bid with travelling distance (to be minimized) equal to 10 and preference (also to be minimized) equal to 4, together with another bid with travelling distance equal to 15 and preference equal to 2. Since the Seller does not know the buyer's weights on travelling distance and preference, neither of these two bids dominates the other.

The seller's challenge is in how to schedule its engineers effectively, though only a simple strategy will be considered in this project.

Naturally, most of the jobs are serviced by the seller in the same region as the buyer. However, better schedules could result in cross-regional services. The challenge in this work is to define

a mechanism for generating good solutions to BT's dynamic workforce scheduling problem. Special attention is paid to the communication protocol among the agents. We want to design a negotiation protocol that supports different optimization algorithms and general advanced heuristic methods.

2.2 A formal definition of the business model

It is essential for the management to formalize their business model. This will allow them to review and improve their current operations. It also enables all parties in the project to verify whether the computational model meets the business specifications. In this section, we provide a rigorous description of the business model. Details of the model may not be needed for the negotiation protocol, but they allow others to evaluate and re-implement our system.

***Job information*:**

Each job is described by a tuple:

(Loc, Sk, D, SD, P)

where Loc is a location, which describes the coordinates (x, y) of the job. Sk is the skill required, which ranges from 1 to 9 (which is not a rank order). D is the duration of the job, in terms of number of days. SD is the starting day, which could be 1 to MaxD, where MaxD is the number of days that we plan ahead (MaxD was set to 7 in our experiments, i.e. to plan 7 days ahead). P is a price, which relates to the value or importance of the job (this may be determined by the manager).

***Engineer information*:**

Each engineer is described by:

(Loc, LSP, SA, OT)

where Loc is a location, which describes the coordinates (x, y) of the engineer. LSP is a list of (*skill*, *preference*) pairs, covering all skills. Each pair describes the preference of this engineer on jobs that require the specific skill. Preferences range between 1 and 9, with 1 meaning most favourable and 9 meaning least favourable. SA is the *surplus availability* of the engineer, which is a list of the days on which this engineer is available. OT is a Boolean variable on overtime availability (0 means no overtime is allowed by this engineer, 1 means overtime allowed).

The problem of each buyer and seller was formulated as an *open constraint*

optimization system, Tsang (2002). An open constraint optimization system is a constraint satisfaction problem, Tsang (1993), where some constraints are not entirely within the control of the problem solver itself. Such constraints cover variables which values are assigned by other solvers. This means to check whether such constraints are satisfied, a problem solver has to negotiate with other solvers. It is therefore an open system, as opposed to a closed one, where all constraints are within the solver's control. (The readers should not confuse this with the Open Constraint Satisfaction Problem defined by Faltings and Macho-Gonzalez (2003), where the number of variables may change dynamically.)

The Buyer's Model:

The problem of buyer *b* can be formulated as an open constraint satisfaction model:

$$(Z_b, D_b, C_b, E_b, f_b, Ag_b, EtA_b, CP_b)$$

where

Z_b = {s[1], s[2], …, s[n_b], p[1], p[2], …, p[n_b], d[1], d[2], …, d[n_b]}, where n_b is the number of jobs that *b* has, s[*i*] represents the service seller that b appoints to do job *i*, p[*i*] represents the preference and d[*i*] represents the distance for serving job *i*, which are proposed by the service sellers and accepted by the buyer;

Ag_b is the set of seller agents who *b* has contact to;

D_b is a function that defines the domain of the variables in Z_b, as in constraint satisfaction, Tsang (1993). For all *i*, D_b(s[*i*]) = Ag_b plus ϕ, which means s[*i*] could be assigned one of the seller agents, or assigned no seller at all (which is represented by the value ϕ); D_b(p[*i*]) = [0..9], which means p[*i*] could be assigned a value 0 to 9, with 0 meaning the job is not served, 1 to 9 are preferences in the service. For all distance variables d[*i*], D_b(d[*i*]) = **R**;

C_b represents a set of internal constraints, which is an empty set in this case, i.e. there are no constraints on what value *b* assigns to the variables;

E_b = { E_b(s[*i*], p[*i*], d[*i*]) | i = 1.. n_b }, where E_b(s[*i*], p[*i*], d[*i*]) is a constraint on the values of s[*i*], p[*i*] and d[*i*], restricting the values that they can take simultaneously; the values of p[*i*] and d[*i*] are to be determined by external agents, s[*i*] indicates the seller that *b* assigns job *i* to; it is assigned by *b*, depending on the bids by the sellers;

f_b is the objective function for *b*. It is a multi-objective function;

$$f_b = (reve_b, failure_b, pref_b, dist_b, comm_b)$$

- Maximize $reve_b = (\Sigma_{i,\, p[i]>0}\ \text{Price}[i])$, which is to maximize total revenue, where Price([i]) is a constant;
- Minimize $failure_b = (\Sigma_{i,\, p[i]=0}\ 1)$, which is to minimize the number of unassigned jobs;
- Minimize $pref_b = (\Sigma_{i,\, p[i]>0}\ p[i])$, which is to minimize the total preference;
- Minimize $dist_b = (\Sigma_{i,\, p[i]>0}\ d[i])$, which is to minimize total distance travelled;
- Minimize $comm_b$ = number of communications.

The buyer b attempts to maximize its utility:

$$\text{Utility} = k_1 \times reve_b - k_2 \times failure_b - k_3 \times pref_b - k_4 \text{ f}\times dist_b - k_5 \times comm_b + \text{Trade}$$

where the weights k_1 to k_5 are given by the manager; Trade is income from other buyers – payment to other buyers for contract-release;

EtA_b is the mapping from each external constraint E_b to a seller agent; i.e. $EtA_b(E_b(s[i], p[i], d[i]))$ returns a value in Ag_b; this indicates that the values of p[i] and d[i] are to be determined by all seller agents (Ag_b) in communication with b;

CP_b is the communication protocol. Here we assume the following protocol:

1. The buyer b sends a set of invitation to bid to seller s; each invitation is

   (Job_ID, Job_information)

   where Job_ID is the identity of the job, and Job_information is a tuple as defined above:

   (Loc, Sk, D, SD, P)

2. The seller s sends a set of pairs of values to b for instantiating (p[i], d[i])
3. The buyer b offers s a contract, which comprises a pair of p and d values
4. The seller s accepts the contract (and commits its resources) or declines the offer, in which case, go back to Step 3 (where b could offer a contract to another seller)

Once variables p (preferences) and d (distances) are communicated between b and s, t hey can be seen as *contracts* between the buyer and the seller.

The Seller's Model:

The problem of seller s can be formulated as a dynamic open constraint satisfaction

model[2]:

$$(Z_s, D_s, C_s, E_s, f_s, Ag_s, EtA_s, CP_s)$$

where

Z_s = {e[1], e[2], …, e[*N*], p[1], p[2], …, p[*N*], d[1], d[2], …, d[*N*]}, where *N* is the total number of jobs that *s* has been invited to bid for and *s* is still in contention (i.e. the buyer has not yet assigned the job to another seller); e[*i*] represents the engineer that is assigned to do job *i*; p and d represents preferences and distances as defined in buyers;

Ag_s is the set of buyer agents who *s* has contact to;

D_s is a function that defines the domain of the variables in Z_s, as in constraint satisfaction, Tsang (1993). For all *i*, D_s(e[*i*]) = the set of engineers plus ϕ, which means e[*i*] could be assigned one of the engineers, or assigned no engineer at all (which is represented by ϕ); D_s(p[*i*]) = [0..9], which means p[*i*] could be assigned a value 0 to 9, with 0 meaning the job is not served, 1 to 9 are preferences in the service; For all distance variables d[*i*], D_s(d[*i*]) = **R**; default values for p and d are 0, which means no engineer is assigned to job *i* until commitment is made (by *s*);

C_s represents the internal constraints that governing the feasibility of the engineers doing the jobs; this involves the availability and skills of the technicians;

E_s = { E_s(e[*i*], p[*i*], d[*i*]) | i = 1.. *N* }, where E_s(e[*i*], p[*i*], d[*i*]) is a constraint on the values of e[*i*], p[*i*] and d[*i*], restricting the values that they can take simultaneously; the values of e[*i*], p[*i*] and d[*i*] are proposed by s, to be approved by the buyer;

f_s is the objective function for *s*. Associated to each job *i*, Price([*i*]) is a constant given by the Buyer. f_s is a multi-objective function:

- Maximize jobs done JD = ($\Sigma_{i,\ p[i]>0}$ 1)
- Minimize distance travelled DT = ($\Sigma_{i,\ p[i]>0}$ d[*i*])
- Maximize load balancing LB = – ($\Sigma_{i=1,N}$ (free_time[*i*]- Mean_free_time)^2)/N where N is the number of engineers
- Minimize redundancy RD = ($\Sigma_{i=1,N}$ (free_time[*i*]/time_available[*i*]))/N

The seller s attempts to maximize its utility:

$$\text{Utility} = k_1 \times JD - k_2 \times (DT)^2 + k_3 \times LB - k_4 \times RD + CR$$

[2] The seller's problem does not have to be formulated as an open constraint satisfaction system. It can be formulated independent of the buyer's model. Here it is formulated as a dynamic scheduling model, with new variables being created when the seller receives requests from the buyer, and variables removed when the buyer does not select this seller.

where the weights k_1 to k_5 are given by the manager; CR is the income from buyers for contract-release.

EtA_s is the mapping from each external constraint E_s to a buyer; i.e. $EtA_s(E_s(e[i], p[i], d[i]))$ equals the service buyer for job i

CP_s: see E_b in the Buyer's model

The Manager (Centralized Coordination):

The objective is to balance failure of all domains. In this research, the following objectives are considered:

- Minimize the average failure rate: $(\Sigma_{b \in Buyer,} FR_b) / NB$
  where $FR_b = failure_b / n_b$, where $failure_b$ is the number of failures for buyer b and n_b is the number of jobs that b has, NB is the number of buyers;
- Minimize the total distance travelled in all jobs done;
- Minimize the average preferences of all jobs done;
- Maximize failure-imbalance = $(\Sigma_{i=1,Nb} (failure_i - Mean\_failures)^2)/Nb$
  where Nb = number of buyers in the system.

The Manager's overall objective is a multi-objective function. As a decision support tool, the manager (program) should attempt to produce a Pareto set of solutions for the end-user (human user, whose job is to look after the company's overall interest in this problem). This can be done by giving the buyer/seller agents different sets of weights for the different buyers and seller measures (the k_i's mentioned above). For example, the k_i's can be used to empower individual buyers to increase their bargaining power so as to reduce their failure rates. The manager may ask the agents to:

- Schedule from scratch; or
- Improve on the previous schedule

3 The RECONNET protocol for BT's problem

This work is loosely based on, and extended from the contract net protocol, a fundamental protocol for distributed artificial intelligence, Davis and Smith (1988), FIPA (2002), Smith (1980). Some modifications are motivated by BT's workforce scheduling problem:

1. In BT's problem, each agent looks after its own interest. They work with each other in order to achieve their common goals (of getting the jobs done).
2. BT's problem is an optimization problem. The operation involves millions of Pounds,

and therefore a small percentage in saving could mean a lot in real terms. Therefore, it is worth investing computation time to reduce cost.

3. The problem is a dynamic problem: new jobs arrive at all time, and availability of staff may change. Therefore, there is a limit in how much computation time one can spend on scheduling.
4. There are hundreds of jobs and engineers in each region. Each job could potentially be served by most engineers, though not all of them are good matches (for example, due to travelling distance). If all possible contracts were communicated, the amount of communication would be impractically large. Therefore, we must attempt to reduce communication.

Given points 2 and 3 above, local search would be a good candidate. A retractable contract net protocol, which we call RECONNET (which stands for Retractable Contract Net), that facilitates hill-climbing by the buyers and sellers is designed. Used by the manager in iterations, RECONNET will enable us to implement any meta-heuristic methods such as Tabu Search, Glover and Laguna (1997), Guided Local Search, Mills (2002), Voudouris and Tsang (2003) and other local search methods (e.g. see Glover and Kochenberger (2003), Hoos and Tsang (2006)).

3.1 Building contracts

A ***contract*** is a tuple:

(ID, Seller, Buyer, Preference, Distance)

where ID is the identity of the contract, Seller and Buyers are the agents involved, Distance and Preferences are as defined above.

Following is the mechanism for building initial contracts in RECONNET:

1. **Invitation to bid:** The buyers initiate contract building. The buyers would invite all sellers (or a subset of sellers, if it sees fit) to bid for all the jobs currently available.
2. **Bidding for jobs:** Given an invitation to bid for a job *j*, a seller will find all engineers who are available and qualified to do *j*. Since each qualifying engineer may be at different locations and have different preferences for *j*, the seller builds up a list of (preference, distance) pairs. To reduce the volume of communication, the seller will only send to the buyer contract proposals that do not dominate each other. In other words, if engineers A and B can both serve job *j*, but A is closer to the location of *j* and

has higher preference than B, then the seller will not bid for a contract with B serving *j*.

3. **Making offers to sellers:** After receiving bids from the sellers, the buyer will build a priority queue for each job. The bids are ordered by their contributions to the buyer's weighted objective function. Jobs are then offered to the proposed contracts that are at the heads of priority queues. In a model, in every round, the buyer makes an offer on every job for which a contract has yet to be secured. This does not prevent the buyers from adopting more complex strategies. When the buyer makes an offer to a seller, the offer is binding. That means if the seller accepts the job, the buyer is not allowed to cancel it and offer the job to another seller.
4. **Acceptance of offers:** In this project, we assume that the seller will accept an offer as long as an engineer is available. When a seller receives multiple contracts that require the same engineer at the same time, it will pick the offer that maximizes its objective function. This simple strategy can be replaced by a more sophisticated algorithm in the future (for example constraint satisfaction techniques and iOpts, Voudouris, et al. (2001)), in order to maximize the chance of doing more jobs (based on probabilities and past experience). For example, a seller may consider fairness amongst the engineers.

When new jobs arrive dynamically, Steps 1 and 2 are invoked. Contracts are being built through repeated steps 3 and 4. At any time, the buyer maintains the status of all the jobs. For example, Table 1 shows a snap shot of the status of four jobs.

Table 1 shows that two bids have been received for Job 1. The offers to both bids have been declined. Contracts have been secured for both jobs 2 and 4. A contract has yet to be secured for job 3, which has one bid left to be unexplored. At this point, the buyer may invite further bids for job 1. The sellers may bid contracts that were dominated by Bids 1 and 2, or contracts that become available due to change of engineers' availability.

3.2 Contract-release

To facilitate hill-climbing, a *contract-release* mechanism is introduced. This is a mechanism for one service buyer to release its contract to another buyer, to enable to latter to complete jobs with higher utility (according to the manager's amalgamation functions). Suppose buyer *s* has a job *j* that cannot be served by any seller given their current commitments. With information provided by the sellers, *s* may offer to pay another buyer to release its contract, and therefore releasing the engineers to do *j*.

A contract-release is a tuple:

((ID, Seller, Buyer, P, D), (*CR_ID*, *Owner*), *Cost*)

where (ID, Seller, Buyer, P, D) is a contract that the seller offers to the buyer who invites bidding. *CR_ID* is the identity of another contract, which must be released before this bid can be implemented. *Owner* is the owner of the contract with *CR_ID*. *Cost* is a cost (which could be 0), from the seller's point of view, of changing the contract. A cost is required if, by changing from the released contract to the new contract, the seller's utility is reduced. For example, the new contract may require more travelling by the seller's engineers.

To complete a contract-release, the Buyer must request the *Owner* to release its contract. It will do so by offering compensation to *Owner*. The compensation is determined by the gain in completing the contract, based on the Buyer's objective function, less the compensation requested by the seller.

Before *Owner* accepts this request, it will look for an alternative contract. This may involve (a) making an offer to the next bid in the priority queue, or, in the case of the queue being reduced to empty, (b) invite further bids from the sellers. A bid from a seller could involve further contract-release. *Owner* decides to accept or reject a contract-release request with the following consideration:

- Releasing the contract for a (typically) inferior contract involves a cost to Owner.
- Owner will accept the contract-release if compensation received covers the cost of switching to the next available bid.

Since every request for contract-release requires compensation, the chain will not continue indefinitely. The contract-release mechanism is shown in Figure 1. For example, suppose Buyer 1 cannot secure a contract for a particular job J. It may get a bid from Seller X, which states "I have an engineer who is committed to job J' with Buyer 2; if Buyer 2 releases this contract, I can do job J with a travelling cost of 18 miles, and a preference of 4; I would also require a compensation of 21 points to me". Buyer 1 would translate this into its own cost, using the parameters that it is given by the Manager. Suppose this translated to a cost of 170, and the utility of competing job J is 390. Then Buyer 1 may make a request to Buyer 2 with a compensation offer of, say 130, to release the contract. Exactly how much Buyer 1 offers to Buyer 2 depends on the algorithm that Buyer 1 uses (e.g. see Gosling et al 2006 and Jin and Tsang 2006). Buyer 2 will evaluate the offer by assessing the cost of securing a new contract for job J', which Seller X can no longer serve. Buyer 2 may have to invoke further contract release chains. If Buyer 1's offer of 130 is higher than the cost of revising the contract, then

Buyer 2 will inform Buyer 1 and Seller X its acceptance of the contract-release offer. Since the benefit that Buyer 2 can offer to other buyers must be less than 130, the contract-release chain will terminate.

3.3 Decisions in contract-release

A buyer *s* may receive more than one potential release contract for completing a job. Each of these release contracts is owned by a buyer. This owner could be another buyer, but it could also be *s* itself. Which release contract should *s* choose? We assume that *s* will always favour contract-release that involves itself. This is because releasing a contract by itself tends to be the cheapest. Besides, by knowing the availability of options in its other jobs, *s* has a better chance of completing the contract release chain. Consuming too many rounds of communication on one contract means having less time for other contracts (given that the whole system runs for a limited amount of rounds). Picking a contract that the buyer itself owns saves communication rounds and increases the chance of completing the chain.

If a job requires more than one day to complete, it is possible that it can only be done by releasing two or more contracts. This will require the buyer to build a tree, as opposed to building a chain of contract-releases. This becomes more complex. A buyer will have to heuristically decide how much to pay each other buyer for a contract-release. Besides, since contract-release offers are binding, the buyer will have to pay others who have released their contract even if the whole tree of release cannot be completed (i.e. this buyer pays the contract-release compensations to other buyers for no return) We have decided to allow only one contract-release per job at this stage because, from both local and global point of view, the chance of benefiting of completing one job by releasing two is not high.

4 ASMCR, an implementation of RECONNET

ASMCR, which stands for Automatic Synchronising Multiple Communication Rounds, is a system that is applied to BT's workforce scheduling problem. It implements the RECONNET protocol for reallocation of jobs. It is written in a way that allows easy adaptation to other problems.

ASMCR was written in Java and makes use of WebLogic, a commercial server platform that supports message passing between processes. Specifically the software uses JMS (Java Messenger Service) for communication. Note that JMS handles messages as broadcasts, but that is just an implementation issue and should not be confused with our logical decision

mentioned in Section 3.1 point 1. If every seller agent sends to the buyers all the available contracts, the amount of communication would be very high. In ASMCR, the seller agents only offer non-dominated contracts, i.e. contracts that are not dominated (in Pareto sense) by other contracts by the same seller.

4.1 Overall Control in ASMCR

The manager's task is to find a Pareto set for the user. There are many ways to conduct the search. In this project, we assume that the Management is responsible for finding the Pareto set. This leaves the sellers and buyers standard optimization problems.
The following control strategy is used in ASMCR, Figure 2.

By providing the agents with different sets of Fx, the manager can make sure that it gets a set of different assignments to make a Pareto set of solutions. In practice, human users like to have control over the system. Therefore, they are given the authority (and responsibility) to choose the solution.

4.2 Job Allocation Procedure in ASMCR

Section 3.1 describes how a contract is built for a job. In this section, our focus is on the agents' control strategies.
Contracts are built through a repeated loop of communications between buyers and sellers. Each round of communication is divided into four phases. The loop terminates when all buyers are satisfied with their contracts for all their jobs or the maximum number of iterations have been reached, see Figure 3.

Communication between the agents is synchronised in ASMCR. The ordering of the four phases is subtly important. The arrangement in Procedure Control_for_Jobs_Allocation (Figure 3) allows a contract or contract-release to be completed within one round.
A technical point needs clarification here. Before buyer X accepts a contract-release from buyer Y, X has to secure a contract to cover the job J served by the released contract. It is possible that buyer Y holds the next best contract to cover job J. If X makes a contract-release request to Y, a cycle would be formed. Cycles are undesirable because contract-release requests are binding offers. That means, once Y has made a request to X, it has to wait for X's reply. Care has been taken in ASMCR to detect cycles in contract-release requests. We have

also imposed a limit to how long a buyer will attempt to fulfil a contract-release request before it gives up.

It is worth elaborating the point that by adjusting the Fx, the Management may conduct a version of guided local search. The manager can reduce the number of unassigned jobs by modifying the cost functions given to the buyers, which could involve increasing incentives, which will increase the length of a contract revision chain.

ASMCR shares the problems with other local search methods. Being an incomplete search, optimality is not guaranteed for its solutions. It is also difficult to know when to stop. The question of when to terminate the search depends on processing power and the desirable response time for the system.

5 ASMCR for BT's distributed workforce scheduling problem

To test its feasibility, ASMCR was tested on real sized data. ASMCR took 5 to 15 minutes to generate a complete plan, and conducted 50 rounds of communication. Like most hill-climbing algorithms, the more time (in terms of rounds) ASMCR is allowed to hill-climb, the more chance it has in finding better or optimal solutions. In a typical run, initial solutions were generated after about 10 rounds. The rest of the rounds contributed to hill-climbing.

For testing ASMCR, we generated random prices between 0.5 and 3.0 for the jobs. We used 7 buyers and 7 sellers, with 150 to 300 jobs and 150 to 300 engineers each. This is within the size range of real problems. To stress-test the algorithm, most of the experiments were generated with far more jobs than engineers.

ASMCR has been tested thoroughly using randomly generated problems. The usefulness of hill-climbing was confirmed. Figure 4 shows a scenario with three regions. Figure 4 (a) shows a situation with three jobs and two engineers. Each engineer was assigned to a job within the same region. A job in the rightmost region was not served. Suppose a new engineer in the leftmost region. When the costing is right, ASMCR was able to reschedule the assignments to serve all the three jobs, as shown in Figure 4 (b). This was only possible when contract-release was enabled.

Experiments also confirmed that by changing the weights to the multiple objectives, the manager can reduce the number of jobs not served, travelling distance and preference (lower value in preference means better service quality).

Table 2 provides an example ASMCR output. The figures were arrived at by averaging the

results of 50 experiments for each parameter set. The configuration parameters used are shown in the lower half and directly relate to those described in Section 2.2.

Table 2 shows a baseline result with its parameters and the effect of adjusting those parameters. For instance by increasing the buyer distance penalty ($dist_b$) from 0.01 to 0.1, a lower contract distance (reduced from 35.29 to 34.74) can be seen in column 3. This change has a cost: the contract preference has risen (from 4 to 5) and the number of completed jobs has been slightly reduced (from 20.93 to 20.64). Columns 4 and 5 show the effects of increasing the weights for buyer preference and sellers' travelling distance, respectively. Desired results were achieved as expected. The results show that ASMCR enables users to generate schedules of different quality by adjusting the weights of the multi-objectives.

The model described in this paper is more general than the actual business set up. (That is why random problems, as opposed to real data, were used for testing.) For example, job prices have yet to be fully explored. ASMCR is a highly configurable system. This enables the system to be adapted to cope with BT's demand. Given its flexibility and efficiency, ASMCR could potentially be developed further to replace the current operational system.

6 Related Work

Contract Net Protocol (CNP) was originally proposed in (Smith 1980) as a high level communication and control mechanism for distributed problem solving. Since then, it was widely adopted and acknowledged in both industry and academics in various fields, for example see Parunak & Van Dyke (1987), Davis & Smith (1983), Ramamritham & Stankovic (1984), Fischer et al (1996), Ouelhadj et al (2003) and Sandholm (2000). In particular, it was heavily studied in distributed scheduling and multi-agents negotiation system.

In the original paper, CNP was used to solve problems from cooperative environment. With the demands from Internet, electronic markets and related fields, recent researches are now focusing on CNPs in self-interested agents (Sandholm 2002) (Akinee 2004), which is a hot topic among distributed artificial intelligence community. However, when CNP is applied to self-interested agent system, some major limitations must be noted:

- The contract in CNP is binding, which means the parties in a contract must devote themselves to the contract without intentionally breaking it. In CNP, a contract can be cancelled by an agent sending a cancel message from the contractor to a contractee. In effect, the contractor may cancel a contract at anytime even if contractee has already

invested a substantial amount of effort into the contract. This arrangement is viable when the agents are cooperative to a common goal. However, it would not be appropriate for self-interested agents.

- When CNP is used in a dynamic environment, where events are not known in advance, the sequence of the events happening will have significant impact on the final result. For example, in BT's dynamic scheduling problem above, new jobs emerge from time to time. At any point of time, a schedule must be available (so that the engineers know which job to attend). Given a schedule *S*, the sequence in which the new jobs become available for biding will affect the result of *S*. When new jobs emerge, repairing *S* is preferred to rescheduling from scratch whenever possible. This is because the engineers may have already prepared for the jobs allocated to them in *S*. Rescheduling from scratch does not help the course of staff empowerment, which is the motivation of this project.
- When agents are self-interested, they may manipulate the bidding in CNP to achieve their maximum benefit. Also, agents normally only have bounded rationality (Simon 1957), which means the decisions (accept/reject bids, choose bids, etc.) they made can only be local optimal. The system's overall payoff may be badly defected by individuals' behaviours. The original CNP cannot readily handle these situations. In RECONNET, contract release facilitates exchanges of jobs. Thus, conflict of interest is handled through a market. As long as the rules are set appropriately, there is a higher chance of generating efficient schedules in a market (Sunder 2004). This project is to define the rules for BT's market.

To address these problems of CNP, various extensions have been proposed. Here we limit ourselves to proposals related to RECONNET. We shall highlight the difference among them as well as their advantages and disadvantages.

In (Fischer et al 1996), a CNP capable of decomposing tasks into sub-tasks is introduced. Both tasks and sub-tasks can be bid separately through auctions. It was a good attempt to attack the problem of contract binding. The technique was useful to domain in which tasks can be decomposed. Auction is an option for biddings in RECONNET.

In (Sen & Durfee 1994, 1996, 1998), decommitting among cooperative agents was studied in a meetings-scheduling domain. The study mainly focused on the commitment to conflicting resources within an agent when bidding. It pointed out that different commitment strategies (commit or not commit) have different effects on different problems. Thus they proposed an adaptive commitment strategy based on the analysis of the resources and events. It used

multistage negotiation protocol for negotiation purpose during the scheduling process, which allowed multiple rounds of negotiation before an agreement is reached between the negotiating agents. In (Sen & Durfee 1996), cancellation and rescheduling of contracts were discussed. Contracts that have higher priority can 'bump' lower priority contracts, which in turn have to be rescheduled later. The cost of cancellation and rescheduling were calculated based on a globally beneficial strategy (from a system perspective) or a locally beneficial strategy (from an individual perspective), which determined whether cancellations could occur. The mechanism used here was similar to RECONNET, except that RECONNET demands compensation for cancellation to protect the interest of the individual agents in self-interested environments. Also, the mechanism was not explicitly used for local search, while RECONNET does.

Contingency Contract is proposed in (Raiffa 1982), in which the obligations of the contract are made contingent on future events. Sandholm & Lesser (2001) noted that contingency contract had certain drawbacks. For one, in real situations, not all possible future events may be known beforehand. Thus contingency contracts may not be used optimally. Besides, the system becomes intractable as the number of future events increases. Most importantly, self-interested agents could manipulate negotiations when they encounter events unbeneficial to them. These drawbacks make contingency contract not favourable in self-interested environment.

Akinee (2004) introduced a two-phase negotiations mechanism to CNP. The bidding for a contract was separated into *PreBid* and *DeffinitiveBid* phases. Agents could change their biding prices strategically during the phases to increase their possibility of winning and help the contractor to find a better contractee with more information. However, it does not allow cancellation of contracts, even though it partly consumed the cancellation internally (between the two phases, changing of a bidding price of a *PreAccepted* winner could be seen as a kind of cancellation). Therefore, as in the original CNP, it could not easily respond to future events. It could be argued that the more phases there were, the better result could be expected. However, in normally circumstances, the negotiation process is limited by time, which makes it not favourable in time critical applications such as BT's workforce scheduling problem.

Sandholm & Lesser (2001, 2002) proposed the Leveled Commitment Protocol (LCP) to avoid the drawbacks of contingency contracts. Agents in a contract can release or decommit its contracts by paying a penalty to the other party. The idea of cancellation with payment is shared by RECONNET. Three games derived from different contracting settings were analysed. It claimed that cancellation with penalty could perform at least no worse than full-commitment CNP, and could increase the expected payoffs of both contract parties. A

similar analysis could be done for RECONNET.

The focuses of RECONNET and LCP are quite different. In RECONNET, we try to use a swap-like mechanism to facilitate local search in the space of schedules. Whenever swap happens, all parties involved will benefit from it. Thus we can hill-climb to another better solution until find a local optima. Although Sandholm & Lesser (1995) suggested agents carrying out their tasks separately can exchange contracts to improve their own utility, it was not widely used to improve the whole system's profitability and efficiency. LCP claimed to be able search and backtrack in the contract space in a systematic way (global search). While this is true, it was obviously not tractable for large scale applications like BT's problem here (which have tens of thousands of jobs to schedule). In either way, search will be costly because of penalties, but RECONNET is supposed to be less costly compared to LCP.

RECONNET and LCP take different approaches to tackle the problem of manipulation by self-interested agents. LCP, from the game theory point of view, proved that there was always at least one Nash equilibrium for LCP, which meant no matter how individuals manipulated, they could not improve their payoff if other individuals pertain to their strategies. However, since LCP's proof was based on Game Theory (to find Nash equilibrium), it assumed perfect information for every party of a contract. In real situations, such as BT's scheduling problem, perfect information is quite often not available. For example, in an auction, one does not normally know the other bidders' bottom line. RECONNET does not assume information about other agents' utilities. It attempts to use the market's *invisible hand* to find locally optimal schedules. Besides, it tackles a multi-objective optimization problem. The manager is given the task of finding a Pareto set of solutions by means of setting the weights for the multiple objectives for the individual agents. Individual can only achieve their best payoff within the manager's weights definitions.

Penalties must be calculated for both RECONNET and LCP. This is a nontrivial problem, especially under incomplete information (which is the case in BT's scheduling problem). In LCP, penalties must be defined before a contract is made binding. In the same authors' earlier work, TRACONET (Sandholm & Lesser 1995), local optimisers were used to calculate local "marginal costs" (which define the penalties) before bidding starts. In (Sandholm et al 1999), algorithms to compute the optimal contract in LCP were provided. In contrary, RECONNET's penalties (prices for contract release, to be precise) are computed dynamically according to the current situation. The exact algorithm used in RECONNET is left open on purpose. In order to support staff empowerment, pricing of contract release may be left to the user in the operational system.

7 Future Research

This project lays the foundations of an architecture that enables hill-climbing in building contract nets. Following are some of the directions that are under investigation:

1. Adjustment of parameters

   The Manager generates one solution at a time, by supplying the buyers and sellers with a set of parameters for amalgamation of objectives. Okabe (2004) defines a set of metrics for measuring the quality of a Pareto Set. Given a Pareto Set of solutions, the Manager would like to enhance the Pareto Set by supplying the Buyers and Sellers with a new set of parameters. With workforce scheduling being a recurrent problem, we attempt to model (using machine learning methods) the relationship between the variable space and the objective space. This enables us to enhance the Pareto Set in an intelligent way.

2. Scheduling

   The Sellers are relatively passive under the current system. This does not have to be, and in fact, should not be the case. A seller should be able to improve its performance, as defined by the objective function that it is given. For example, the current stipulation is that the seller will accept offers with maximum utility (a greedy strategy). It may benefit from delaying acceptance of offers for engineers who can serve many jobs, and committing engineers who has fewer options. This resembles the smallest-domain-first principle in constraint satisfaction, Smith and Grant (1998), Tsang (1993). Besides, jobs could sometimes take more time to complete. Schedule-repairing is also high on our research agenda. Guided local search (Voudouris and Tsang 2003) is a candidate algorithm for repairing.

3. Real life concerns

   The work described above is only part of a larger picture. In practice, the manager has to deal with dynamic scheduling, where staff may become unavailable, jobs may be added or cancelled. Strategic planning is also needed, for example, forecasting of demands could be used for planning the workforce.

8 Conclusions

This paper reports a general architecture and a negotiation protocol to facilitate employee

empowerment in workforce scheduling. The aim is to arrive at all-win schedules for the top management and service buyers and sellers, each of whom is given an opportunity to look after its own interests. It is tested on BT's workforce scheduling problem, which is modelled as an open constraint optimization system. The overall problem is a multi-objective optimization problem: one attempts to maximize completion rates and service quality and minimize travelling distances. The job of balancing different objectives is given to the manager. This leaves us with single-objective optimization problems to solve at the lower level. We do not assume that the manager knows what the optimal weights are for all the parameters. The proposal is to generate a Pareto set of schedules for the human users.

Standard contract net is a practical strategy in distributed scheduling where agents may have conflicting objectives. Therefore, it is used as our starting point. We have introduced a retractable contract net protocol, RECONNET, which supports hill-climbing in the space of solutions. It is built upon a job-release and compensation mechanism. RECONNET is a general protocol, which supports general multi-objective optimization algorithms and meta-heuristic algorithms.

A system based on RECONNET has been implemented in Java using WebLogic at BT for its workforce scheduling problem. The software, which we call ASMCR, allows the management to have full control over the company's multi-objectives. The manager generates a Pareto set of solutions by defining, for each buyer and seller, the weights given to each of their objectives. ASMCR also gives service buyers and sellers ownership of their problem and freedom to maximize their performance under the criteria defined by the management. ASMCR took 5 to 15 minutes to complete when tested on real-sized problems. It has full potential to be developed to tackle BT's workforce scheduling problem.

Acknowledgement

Edward Tsang was sponsored by a BT Short Term Fellowship, 2004. Tim Gosling was sponsored by a BT-EPSRC CASE Studentship. Wudong Liu is sponsored by a BT Studentship.

References

[1] Aknine, S., Pinson, S. & Shakun, M. F. (2004). An extended multi-agent negotiation protocol. *Autonomous Agents and Multi-Agent System*, 8(1):5–45

[2] Borrett, J.E. & Tsang, E.P.K. (2001). A context for constraint satisfaction problem formulation selection, Constraints, Vol.6, No.4, 2001, 299-327

[3] Davis, R., and Smith, R.G. (1988). "Negotiation as a metaphor for distributed problem solving". In A.

Bond, and L. Gasser, L. (eds.), *Readings in distributed artificial intelligence*, Morgan Kaufmann, 333-356.

[4] Durfee, E.H. (1999). "Distributed problem solving and planning" In G. Weiss, (eds.), *Multiagent systems, a modern approach to distributed artificial intelligence*, MIT Press, 121-164.

[5] Faltings, B. and Macho-Gonzalez, S. (2003). "Open constraint optimisation". In F. Rossi, (eds.), *Proceedings, Principles and Practice of Constraint Programming (CP 2003), Lecture Notes in Computer Science 2833*, Springer, 303-317.

[6] Faltings, B. (2005). A Budget-balanced, Incentive-compatible Scheme for Social Choice. Agent-Mediated Electronic Commerce VI, LNAI, 3435, pp. 30-43.

[7] FIPA (Foundation for Intelligent Physical Agents). (2002*). FIPA Contract Net Interaction Protocol Specification*. http://www.fipa.org/specs/fipa00029/SC00029H.html

[8] Fischer, K., Muller, J.P. and Pischel, M. (1996). Scheduling an application domain for dai. *Applied Artifical Intelligence, An International Journal*, 10:1–33.

[9] Freuder, E.C. (1999). Modeling: the final frontier, The First International Conference on The Practical Application of Constraint Technologies and Logic Programming (PACLP), London, April 1999, 15-21

[10] Glover, F.W. and Laguna, M. (1997). *Tabu Search*. Kluwer Academic Publishers

[11] Glover, F. & Kochenberger, G.A. (ed.) (2003). Handbook of metaheuristics, Kluwer

[12] Gosling, T., Jin. & Tsang, E.P.K. (2006). Games, supply chains and automatic strategy discovery using evolutionary computation, in J-P. Rennard (Eds.), Handbook of research on nature inspired computing for economics and management, Chapter XXXVIII, 572-588

[13] Hoos, H. & Tsang, E.P.K. (2006). Local search for constraint satisfaction, in F. Rossi, P. van Beek & T. Walsh (ed.), Handbook of Constraint Programming, Elsevier, 245-277

[14] Ito, T., Hattori, H., Zhang M. and Matsuo, T. (Eds.), Rational, Robust, Secure Negotiations in Multiagent Systems, IEEE Press, 2005

[15] Jin, N. & Tsang, E.P.K. (2006). Co-adaptive Strategies for Sequential Bargaining Problems with Discount Factors and Outside Options, Proceedings, Congress on Evolutionary Computation (CEC), 7913-7920

[16] Mills, P. (2002). *Extended Guided Local Search*. PhD Thesis. Department of Computer Science, University of Essex.

[17] Okabe, T. (2004). Evolutionary multi-objective optimization. Shaker Verlag Aachen.

[18] Ouelhadj, D., Cowling, P. and Petrovic, S. (2003). Intelligent Systems Design and Applications, chapter *Contract net protocol for cooperative optimisation and dynamic scheduling of steel production*, pages 457–470. Springer-Verlag.

[19] Parunak, H. Van Dyke. (1987). Manufacturing experience with the contract net. In *Distributed artificial intelligence, Research notes in artificial intelligence*, 0268-7526, pages 285–310. Pitman, London.

[20] Prosser, P. (1990). *Distributed asynchronous scheduling*. PhD Thesis. Department of Computer Science,

University of Strathclyde.

[21] Raiffa, H. (1982). *The Art and Science of Negotiation*. Harvard University Press, Combridge, Mass.

[22] Ramamritham, K. and Stankovic, J. A. (1984) Dynamic task scheduling in hard real-time distributed systems. *IEEE Software*, pages 65–75.

[23] Sandholm, T. and Lesser, V. (1995). Issues in automated negotiation and electronic commerce: Extending the contract net framework. In Victor Lesser, editor, *Proceedings of the First International Conference on Multi-Agent Systems (ICMAS'95)*, pages 328–335, San Francisco, CA, USA, 1995. The MIT Press: Cambridge, MA, USA.

[24] Sandholm, T., Sikka, S. and Norden, S. (1999). Algorithms for optimizing leveled commitment contracts. In *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence*, pages 535 – 541. Morgan Kaufmann Publishers Inc.

[25] Sandholm, T. (2000). Automated contracting in distributed manufacturing among independent companies. *Journal of Intelligent Manufacturing*, 11:271–283.

[26] Sandholm, T. and Lesser, V. (2001). Leveled commitment contracts and strategic breach. *Games and Economic Behavior*, 35:212–270.

[26] Sandholm, T. and Lesser, V. (2002). Leveled-commitment contracting: A backtracking instrument for multiagent systems. *AI Magazine*, 23(3):89–100.

[28] Sandholm, T. (2003). "Automated mechanism design: A New Application Area for Search Algorithms". In Rossi, F. (ed.), Proceedings of the International Conference on Principles and Practice of Constraint Programming (CP), 19-36.

[29] Sen, S. and Durfee, E. (1994). The role of commitment in cooperative negotiation. *International Journal on Intelligent Cooperative Information System*, 3(1):67–81.

[30] Sen, S. and Durfee, E. (1996). A contracting model for flexible distributed scheduling,. Annals of Operations Research, 65:195–222.

[31] Sen, S. and Durfee, E. (1998). A formal study of distributed meeting scheduling. *Group Decision and Negotiation*, 7:265–289.

[32] Sim, K. M. and Choi C.Y. (2003). "Agents that React to Changing Market Situations". *IEEE Transactions on Systems, Man and Cybernetics, Part B: Cybernetics*, Vol. 33, No. 2, pp 188-201

[33] Sim, K. M. (2005). "Equilibria, Prudent Compromises, and the "Waiting" Game". *IEEE Transactions on Systems, Man and Cybernetics, Part B: Cybernetics*, Vol. 35, No. 4, pp. 712-724

[34] Simon, H. (1957). "Models of Man", Wiley, New York.

[35] Sunder, S. (2004). "Market as an artefact aggregate efficiency from zero intelligence traders". In Augier, M.E. & March, J.G. (ed.), Models of a Man: Essays in memory of Herbert A. Simon, Cambridge, MA: MIT Press. 2004. 501-519.

[36] Smith, B.M. and Grant, S.A. (1998). "Trying harder to fail first". In H. Prade (eds.), *Proceedings,*

European Conference on Artificial Intelligence (ECAI 98), Wiley. 249-253.

[37] Smith, R.G. (1980). "The contract net protocol: high-level communication and control in a distributed problem solver". *IEEE Transactions on Computers*, Vol.C-29, No.12, 1104-1113.

[38] Thomas, K. W. and Velthouse, B. A. (1990) Cognitive Elements of Empowerment: An 'Interpretive' Model of Intrinsic Task Motivation. Academy of Management Review, Vol 15, No. 4, 666-681

[39] Tsang, E.P.K. (1993). *Foundations of Constraint Satisfaction*. Academic Press

[40] Tsang, E.P.K. (2002). *Constraint satisfaction in business process modelling*. Technical Report CSM-359, University of Essex.

[41] Ursu, M., Virginas, B. and Voudouris, C. (2004), "Distributed resource allocation via local choices: general model and a basic solution". *Proceedings, 8th International Conference on Knowledge-Based Intelligent Information & Engineering Systems (KES2004)*. Wellington, New Zeeland.

[42] Virginas, B., Ursu, M., Owusu, G. and Voudouris, C. (2005). "Intelligent workforce allocation within an agent-based paradigm: central and distributed decision powers". *The IASTED International Conference on Artificial Intelligence and Applications (AIA 2005)*. The 23rd IASTED International Multi-Conference on Applied Informatics, Austria.

[43] Voudouris, C., Dorne, R., Lesaint, D. and Liret, A. (2001). "iOpt: a software toolkit for heuristic search methods". *Proceedings of the 7th International Conference on Principles and Practice of Constraint Programming*. Springer-Verlag, 716-729.

[44] Voudouris, C. and Tsang, E.P.K. (2003). "Guided local search". In Glover, F. (eds.) Handbook of metaheuristics. Chapter 7, Kluwer, 185-218.

[45] Wilkinson, A. (1998). Empowerment: theory and practice. Personnel Review. Vol. 27(1): 40-56

[46] Yokoo, M. and Ishida, T. (1999). "Search algorithms for agents" In Weiss, G. (eds.), Multiagent systems, a modern approach to distributed artificial intelligence. MIT Press, 165-199.

Authors Biographical Notes

Edward Tsang is a Professor in Computer Science at University of Essex. He is also the Deputy Director of Centre for Computational Finance and Economic Agents (CCFEA) and a founding member of the Centre for Computational Intelligence at University of Essex. Edward Tsang has broad interest in business applications of artificial intelligence, including, computational finance, computational economics and constraint satisfaction. Main techniques used included heuristic search, optimisation and evolutionary computation.

Tim Gosling obtained his PhD from University of Essex in 2007. He is now a software expert in the private sector and is a member of the Artificial Intelligence and Games Research Network. His interest is in business applications of artificial intelligence, including games and supply chain management. He is an expert in business modelling, evolutionary computation and reinforcement learning.

Botond Virginas received an MEng in Automation and Computer Science from the Technical University of Cluj Napoca, Romania, in 1990 and an MSc in Computer Aided Environmental Engineering from the University of Manchester in 1995. He holds a PhD degree in Computer Science from Portsmouth University. Between 1997 and 2002 he was a Senior Lecturer in Computer Science at Staffordshire University. Since May 2003 he has been working in the Intelligent Systems Research Centre, on applying agent-based techniques to resource management and more recently to personalisation of next-generation communication services. His research interests include Intelligent Agents, Knowledge Based Systems and Software Engineering.

Chris Voudouris is a senior technology manager at British Telecommunications plc and heading the company's enterprise technologies group. He is also a Visiting Professor at University of Essex and a member of City Associate Board at the Centre for Computational Finance and Economic Agents (CCFEA). Chris Voudouris has won several awards at BT and from organisations such as INFORMS, BCS and the IET. He has broad interest in business applications of artificial intelligence, including scheduling and a wide range of telecommunication applications. Main techniques used include heuristic search, optimisation and agent technology.

Gilbert Owusu is responsible for R&D of next generation systems for resource management in

BT. He has considerable experience in developing enterprise wide software solutions. Gilbert holds a PhD in applied Artificial Intelligence from Leeds Metropolitan University. He is a member of the British Computer Society and a Chartered Engineer.

Wudong Liu obtained his BS in Software Theory and MS in Software Engineering from Wuhan University in 2001 and 2004 separately. He is now a PhD student in University of Essex. His main research focuses on multiobjective optimization, evolutionary computation and their application in industry areas such as management and scheduling. His research is sponsored by a BT studentship.

Tables

Table 1. Sample priority queue for a particular service buyer

	Bid 1	Bid 2	Bid 3	Bid 4	Bid 5	Bid 6
Job1	×	×				
Job2	✓					
Job3	×	×	×	×		
Job4	×	×	✓			
Keys: ✓ means contract secured, × means bids declined						

Table 2. ASMCR Performance

	Base line	**Change in Buyer Distance**	**Change in Buyer Preference**	**Change in Seller Distance**
Results				
Completed Jobs	20.93	20.64	20.71	18.57
Contract Distance	35.29	**34.74**	38.14	**30.92**
Contract Preference	4	5	**2**	4
Control Parameters				
Buyer Revenue ($reve_b$)	5	5	5	5
Buyer Unassigned ($failure_b$)	3	3	3	3
Buyer Distance ($dist_b$)	0.01	**0.1**	0.01	0.01
Buyer Preference ($pref_b$)	0.01	0.01	**0.1**	0.01
Seller Complete (JD)	5	5	5	5
Seller Distance (DT)	0.01	0.01	0.01	**0.1**
Seller LoadBalancing (LB)	0.1	0.1	0.1	0.1
Seller Redundancy (RD)	0.1	0.1	0.1	0.1

Figures

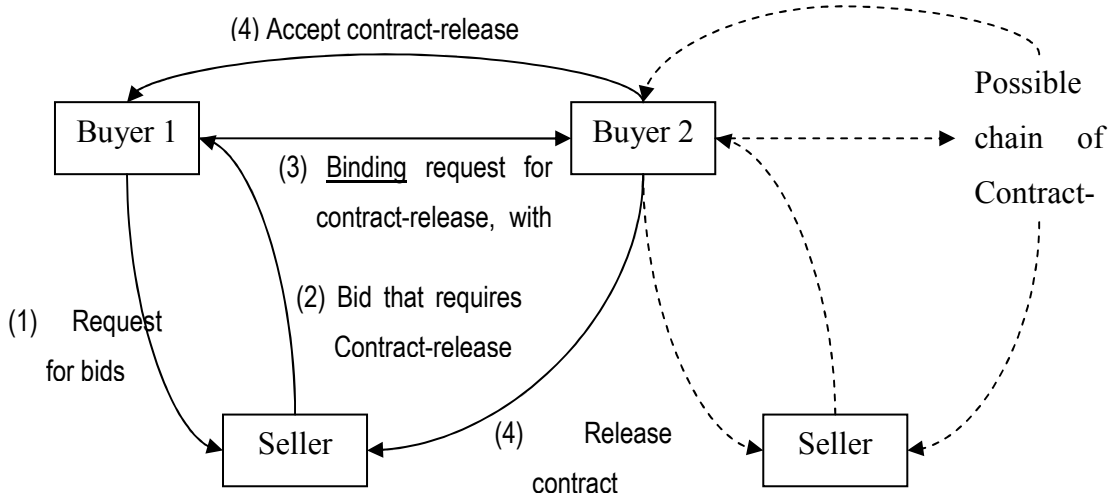


Figure 1: The Contract-release mechanism

Procedure Overall_Control_by_Management

/ the management defines and iteratively revises the cost function for job allocation */*

Repeat

1. Send each agent x (which is either a buyer or a seller) a cost function Fx, which determines how multi-objectives are amalgamated
2. Run **Control_for_Jobs_Allocation**(Fx's)

   / agents may be asked to build on their previous assignments or schedule afresh */*
3. The buyers and sellers submit their final plan to the management
4. The management revises the function

Until all sellers have received an End of Process from every buyer, or a maximum number of iterations

Figure 2: The procedure Overall_Control_by_Management

Procedure Control_for_Jobs_Allocation (Fx's)

/ Fx's is a set of amalgamation functions, one for each buyer and each seller. */*

/ Buyers and sellers build their contracts by allocating jobs to technicians. */*

/ All messages can be empty, with an End of Message marker. */*

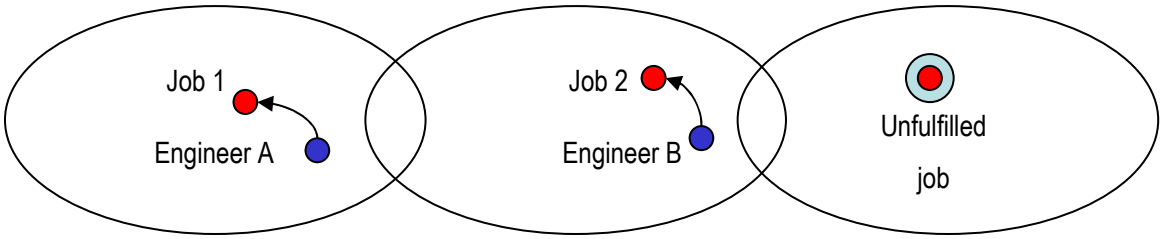
Repeat

1. Each buyer *b* sends a message to each other buyer *b1* (in parallel)
   The message may contain:
   Request for contract-release (binding)
2. Each buyer *b* sends a message to each seller *s* (in parallel)
   The message may contain:
   Request for bids
   Request for bids with possible release of contracts
   Job offers (binding)
3. Each service sellers *s* responses to each buyer who has sent *s* a message
   The message may contain:
   A list of options for each job sent by *s*
   A list of contract-release options for each job sent by *s*
   Accept offer
   Decline offer
4. Each buyer that receives a contract-release request replies to the requesting buyer
   The message may contain:
   Agreement to release contract (binding)
   Decline in contract-release
   Wait for further decisions

Until all sellers have received an End of Process from every buyer, or the search has reached a maximum number of iterations

Figure 3: The procedure Control_for_Jobs_Allocation

(a) Original schedule, with one unfulfilled job in the rightmost region



(b) With a new free engineer, ASMCR rescheduled the jobs using contract-release (released schedules shown in dotted line)

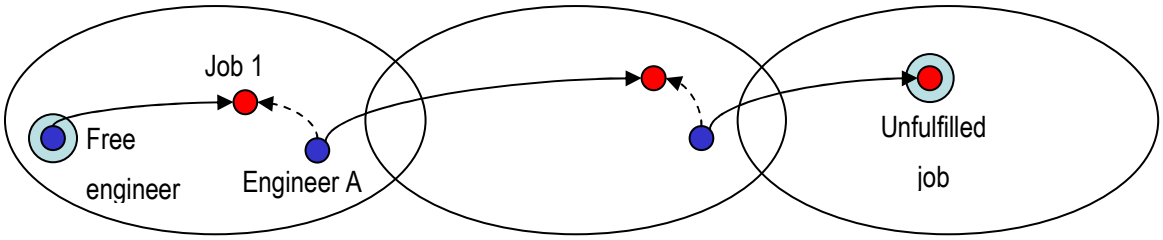


Figure 4: The contract release mechanism enables ASMCR to revise contracts