Proceedings, 2nd Multidisciplinary International Conference on Scheduling: Theory & Applications (MISTA), New York, July 2005, 485-500

Retractable Contract Network for Distributed Scheduling

E.P.K. Tsang[1], T.Gosling[1], B. Virginas[2], C. Voudouris[2] & G. Owusu[2]
Submission to MISTA 2005

[1]Department of Computer Science, University of Essex
[2]BT Research Laboratories

Abstract

This paper is about distributed scheduling where individual agents have potentially conflicting interests. It is motivated by BT's workforce scheduling problem, where 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. In this paper, BT's problem is modelled as an open constraint optimization system.

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, which we call RECONNET, that supports hill-climbing in the space of solutions. It is built upon a job-release and compensation mechanism. RECONNET is a general protocol, which could be used to implement complex meta-heuristic algorithms such as Tabu Search and Guided Local Search.

A system based on RECONNET has been implemented for BT's 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 solution 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 took 5 to 15 minutes to complete when tested on real-sized problems. It has potential to be developed into practical solutions to BT's workforce scheduling problem.

Keywords: scheduling, constraint optimization, multi-objective optimization, meta-heuristic search

1. Introduction

1.1. Distributed scheduling, motivations

In many real life scheduling problems where multiple entities have to work together, centralised scheduling is very difficult. For example:

1. This will be the case when different entities have potentially conflicting interest. Should scheduling be done centrally, independent entities may not inform the central scheduler all the relevant information (game playing).
2. It will also be the case when each entity has many constraints to consider, and these constraints change dynamically. For example, jobs or staff availability may change frequently. In such situations, it may be too tedious to inform a central scheduler all the minor considerations and changes.
3. Human factors may lead to local considerations not being conveyed to the central scheduler. For example, a local manager may take staff emotion into consideration and apply discretion in scheduling his staff.

In such cases, distributed scheduling may be more productive. This paper uses a case study to present a general strategy in distributed scheduling.

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 [2][10][14][18][28][29][30]. In this paper, motivated by real life applications, we study distributed constraint satisfaction problems where agents may have conflicting goals. Work in this area is relatively scarce, but see [4].

1.2. BT's distributed workforce scheduling problem

This research is motivated by BT's workforce scheduling problem, which involves multiple agents with potentially conflicting objectives. In this paper, the problem is formulated as a distributed scheduling problem, which is described below and formalized in the next section [21][22].

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 is 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 distributed workforce scheduling problem

2.1. Overview of the problem

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. Its role is to build a Pareto set of schedules for the end-user to choose from. This is achieved by passing to each buyer and seller an amalgamation function that defines the weights of each agent. The agents may be given different amalgamation functions.

   The manager's challenge is in how to define the amalgamation functions to pass to the buyers and sellers, in order to represent the company's interest. In multiple runs, it attempts to build a "good" Pareto set of schedules. The quality of a Pareto set can be defined in many ways (see, for example, [13] for a good survey). In this project, the control of what Pareto set to produce is left entirely to the user.

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 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 the amalgamation function given by the manager.

[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.

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.

   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 a protocol that supports the implementation of advanced heuristic methods, such as Tabu Search [6][7][8][9] and Guided Local Search [11][12][23][24][27].

2.2. A formal definition of the business model

In this section, we provide a rigorous description of the business model.

***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. 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* [20]. An open constraint optimization system is a constraint satisfaction problem [19] where some constraints are not entirely within the control of the problem solver itself. It is, instead, shared with 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 & Macho-Gonzalez [3], 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 [19]. 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:

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

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:
   (Location, Min_Skill, Duration, StartDay, Price)
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*, they 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 [19]. 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 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 = $-\ (\Sigma_{i=1,N}\ (\text{free\_time}[i]-\ \text{Mean\_free\_time})^2)/N$ where N is the number of engineers
- Minimize redundancy RD = $(\Sigma_{i=1,N}\ (\text{free\_time}[i]/\text{time\_available}[i]))/N$

The seller s attempts to maximize its utility:

[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.

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

Where the weights k_1 to k_5 are given by the manager; Trade 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: $(\sum_{b \in \text{Buyer},} FR_b) / NB$
  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 = $(\sum_{i=1,Nb} (\text{failure}_i - \text{Mean\_failures})^2)/Nb$
  where Nb = number of buyers in the system.

The Manager's overall objective is a multi-objective function. It is not necessary to combine the above measures – this is a job left to the end-user. However, the manager should attempt to produce a Pareto set of solutions. 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 [1][5][16][17]. Some modifications are motivated by BT's workforce scheduling problem:

1. In BT's problem, each agent looks after its own interest. They cooperate 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 is designed. Used by the manager in iterations, RECONNET will enable us to implement any meta-heuristic methods such as Tabu Search and Guided Local Search.

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 have different locations and preferences for *j*, the seller would build 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 a contract for B to do *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 [25][26]), 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. 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 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.

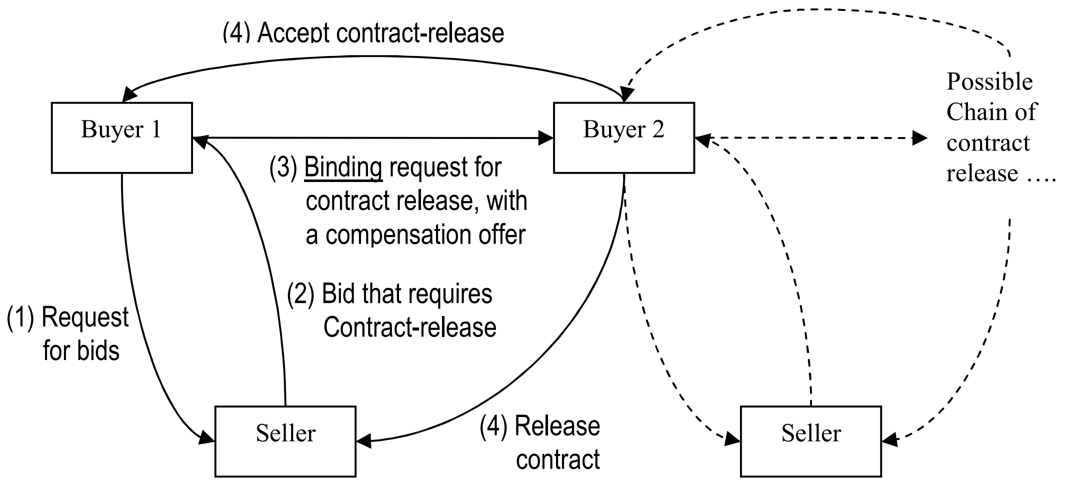
(4) Accept contract-release
Buyer 1
Buyer 2
Possible Chain of contract release ….
(3) Binding request for contract release, with a compensation offer
(2) Bid that requires Contract-release
(1) Request for bids
Seller
(4) Release contract
Seller

Figure 1 The contract release mechanism

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.

```
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
```

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 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.

```
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
```

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 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 2 shows a scenario with three regions. Figure 2 (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 2 (b). This was only possible when contract-release was enabled.

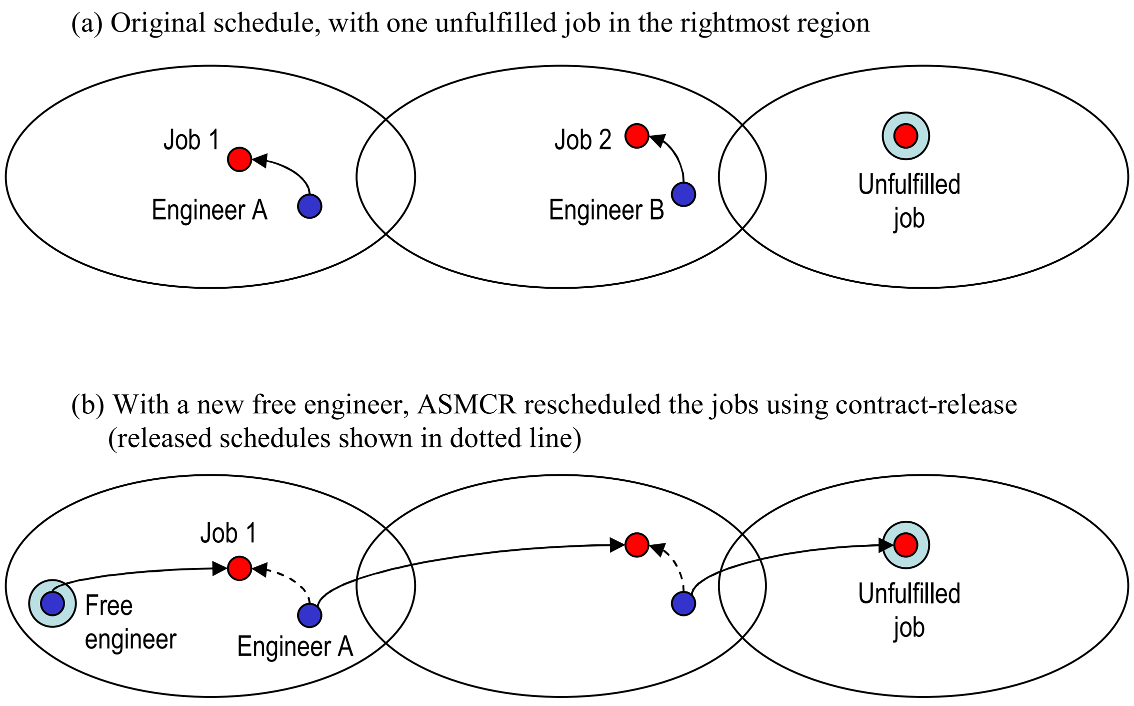
(a) Original schedule, with one unfulfilled job in the rightmost region
Job 1
Engineer A
Job 2
Engineer B
Unfulfilled job
(b) With a new free engineer, ASMCR rescheduled the jobs using contract-release (released schedules shown in dotted line)
Job 1
Free engineer
Engineer A
Unfulfilled job

Figure 2 The contract release mechanism enables ASMCR to revise contracts

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. 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 Redundency (RD)	0.1	0.1	0.1	0.1

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 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' traveling 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 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 into a practical system.

6. 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 worth serious research in the future (in order of importance and promise):

1. Adjustment of parameters – how the manager should adjust the parameters for amalgamation of objectives.
2. The Hill climbing procedure – how the hill-climbing protocol may be improved to maximize utilization of the resources and minimize failure.

3. Scheduling by the seller – the seller is 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 [15][19].

7. Conclusions

This paper is about distributed scheduling where individual agents have potentially conflicting interests. It is motivated by BT's workforce scheduling problem, where multiple service providers have to serve multiple service buyers. Each agent attempts to maximize its utility. In this paper, BT's problem is modelled as an open constraint optimization system. The overall problem is a multi-objective optimization problem: one has 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.

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 that supports hill-climbing in the space of solutions. It is built upon a job-release and compensation mechanism. RECONNET is a general protocol, that could be used to implement complex meta-heuristic algorithms such as Tabu Search [6][7][8][9] and Guided Local Search [11][12][23][24][27].

A system based on RECONNET has been implemented in 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 potential to be developed into practical solutions to BT's workforce scheduling problem.

Acknowledgements

Edward Tsang was sponsored by a BT Short Term Fellowship, 2004. Tim Gosling was sponsored by a BT-EPSRC CASE Studentship. The anonymous reviewers' comments have helped to improve this paper.

References

[1] Davis, R., and Smith, R.G., Negotiation as a metaphor for distributed problem solving, in: Bond, A.,and Gasser, L. (ed.), Readings in distributed artificial intelligence, Morgan Kaufmann, 1988, 333-356

[2] Durfee, E.H., Distributed problem solving and planning, in Weiss, G. (ed), Multiagent systems, a modern approach to distributed artificial intelligence, MIT Press, 1999, 121-164

[3] Faltings, B. & Macho-Gonzalez, S., Open constraint optimization, in Rossi, F. (ed.), Proceedings, Principles and Practice ofConstraint Programming (CP 2003), Lecture Notes in Computer Science 2833, Springer 2003, 303-317

[4] Faltings, B., A budget-balanced, incentive-compatible scheme for social choice, Manuscript, Artificial Intelligence Laboratory, Swiss Federal Institute of Technology, May 2004

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

[6] Glover, F., Tabu search Part I, in Operations Research Society of America (ORSA) Journal on Computing, Vol. 1, 1989, 109-206

[7] Glover, F., Tabu search Part II, in: Operations Research Society of America (ORSA) Journal on Computing, Vol. 2, 1990, 4-32

[8] Glover, F.W. & Laguna, M., Tabu Search, Kluwer Academic Publishers, 1997

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

[10] Luo, Q., Hendry, P. & Buchanan, I., A new algorithm for dynamic distributed constraint satisfaction problems, Research Report KEG-4-92, University of Strathclyde, 1992

[11] Mills, P. & Tsang, E.P.K., Guided Local Search for solving SAT and Weighted MAX-SAT Problems. In Journal of Automatic Reasoning, Special Issue on Satisfiability Problems, Kluwer, Vol.24, 2000, 205-223

[12] Mills, P., PhD Thesis, Extended Guided Local Search, Department of Computer Science, University of Essex, July 2002

[13] Okabe, T., Evolutionary multi-objective optimization, Shaker Verlag Aachen 2004

[14] Prosser, P., Distributed asynchronous scheduling, PhD Thesis, Department of Computer Science, University of Strathclyde, November, 1990

[15] Smith, B.M. & Grant, S.A., Trying harder to fail first, in Prade, H. (ed.), Proceedings, European Confenerence on Artificial Intelligence (ECAI 98), Wiley, 1998, 249-253

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

[17] Smith, R.G., and Davis, R., Frameworks for cooperation in distributed problem solving, IEEE Transactions on Systems, Man, and Cybernetics Vol.11, No.1, 1981, 61-70

[18] Tel, G., Distrubted control algorithms for AI, in Weiss, G. (ed), Multiagent systems, a modern approach to distributed artificial intelligence, MIT Press, 1999, 539-580

[19] Tsang, E.P.K., Foundations of Constraint Satisfaction, Academic Press 1993

[20] Tsang, E.P.K., Constraint satisfaction in business process modelling, Technical Report CSM-359, University of Essex, Colchester, UK, January, 2002

[21] Ursu, M., Virginas, B. & Voudouris, C., 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, September 2004

[22] Virginas, B., Ursu, M., Owusu, G. And Voudouris, C., 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 Twenty-Third IASTED International Multi-Conference on Applied Informatics, Innsbruck, Austria, February 14-16, 2005

[23] Voudouris, C., Guided Local Search - An Illustrative Example in Function Optimisation. In BT Technology Journal, Vol.16, No.3, July 1998, 46-50

[24] Voudouris, C. & Tsang, E.P.K., Guided Local Search and its application to the Travelling Salesman Problem. In European Journal of Operational Research, Elsevier Science Publishers, Vol.113, Issue 2, March 1999, 469-499

[25] Voudouris, C., Dorne, R., Lesaint, D. & Liret, A., iOpt: a software toolkit for heuristic search methods, Proc., 7th International Conference on Principles and Practice of Constraint Programming, Springer-Verlag, 2001, 716-729

[26] Voudouris, C., and Dorne, R., Integrating Heuristic search and One-Way Constraints in the iOpt Toolkit, in Voss, S. and Woodruff, D. (ed.), Optimization Software Class Libraries, Chapter 6, pp 177-192, Kluwer Academic Publishers, 2002.

[27] Voudouris, C. & Tsang, E.P.K., Guided local search, Chapter 7, in Glover, F. (ed.), Handbook of metaheuristics, Kluwer, 2003, 185-218

[28] Yokoo, M., Ishida, T. & Kuwabara, K., Distributed constraint satisfaction for DAI problems, AAAI Spring Symposium on Constraint-based Reasoning, March, 1991, 191-199

[29] Yokoo, M., Weak-commitment search for solving constraint satisfaction problems, Proc., 12th National Conference for Artificial Intelligence (AAAI), 1994, 313-318

[30] Yokoo, M. & Ishida, T., Search algorithms for agents, in Weiss, G. (ed), Multiagent systems, a modern approach to distributed artificial intelligence, MIT Press, 1999, 165-199