Knowledge-Based Distributed Search Using Teamwork

Jörg Denzinger
Department of Computer Science
University of Kaiserslautern
Postfach 3049, 67653 Kaiserslautern, Germany
E-mail: denzinge@informatik.uni-kl.de

Abstract

We present a knowledge-based distribution concept for search problems that offer no natural way to determine several agents to cooperate in finding a solution. Systems based on our teamwork method use four types of agents: Experts and specialists use heuristics to generate results that are possible parts of solutions, referees judge the experts and their results determining the most promising ones and a supervisor collects these promising results to generate new problem descriptions that converge to a solution of the initial problem. The main difficulty of distributed systems, the communication overhead, is dealt with by restricting the work of referees and the supervisor to short so-called team meetings that interrupt the work of experts and specialists.

The competition and cooperation of experts and specialists in this framework allow for synergetic effects that generate better and faster solutions to the search problems. We demonstrate these effects for instantiations of two very different kinds of search problems, automated theorem proving and optimization problems.

Introduction

Methods for traversing very large search spaces, either to find any solution or the best (or at least a very good) solution to a problem, play a very important role in many areas of artificial intelligence. For most of the interesting problems the search spaces are so huge that their systematic construction is not feasible. Even if the development of the search spaces is done in parallel, feasibility is not always achieved. On the other hand, even intelligent pruning of a search space by using a heuristic helps only for part of the instances of a search problem. The question remains, which of the many known heuristics is to choose for a given instance of the problem.

Distributed systems using specialized agents that employ different methods for developing a search space, as for example suggested in [De91], [LL92] or [TSM93], seem to offer a way to combine knowledge-based approaches to search with the possibility of using several

processors or even computers to achieve high computing performances. Nevertheless, there are many different types of search problems that need different distribution concepts.

In this paper we present a knowledge-based distribution concept, the *teamwork method*, for search problems, whose descriptions offer no possibilities for using the known divide-and-conquer based distribution methods, because subproblems to a given problem can not or only very seldom be constructed a priori. These problems resist, naturally, blackboard-based distribution concepts. Examples for these problems are such different ones as automated theorem proving or the solution of NP-hard optimization problems.

Distributed systems designed using the teamwork method use four types of components: experts, specialists, referees and a supervisor. Experts and specialists are the components that work on finding a solution to the given problem. While all experts use common data structures to represent the given problem and their work on it, the specialists are allowed to use any structure and knowledge that may help to solve the problem or aspects of it. Most of the time experts and specialists work independently on the given search problem, using different methods and heuristics. After a certain period of time they stop their work and a team meeting begins. First the work of experts and some of the specialists is judged by referees. A referee judges the progress of its expert or specialist on the given problem and selects its best results. This report is transmitted to the supervisor, that generates a new problem description out of the received data and sends it as new starting point to all experts and specialists. Because there are very often much more experts and specialists than processors available, the supervisor also uses the reports of the referees to exchange experts and specialists with a bad performance on the actual problem.

Teamwork has proved to be quite successful on the class of problems mentioned above. Not only can teams find solutions to a problem faster than the used experts and specialists working alone, but we were also able to solve problems that none of the used components were able to solve working alone. The characteristics of a

teamwork-based system are

- combination of cooperation and competition

- restricted amount of communication between agents

- adaptation of the team to a given problem

- use of all methods and features of sequential systems for a search problem

In this paper we will demonstrate the success of teamwork on two very different kinds of search problems, namely the aforementioned problems automated theorem proving and optimization. As instance of the first area we have chosen equational deduction and as optimization problem the traveling salesman problem. In both problem areas we were able to obtain synergetic effects that resulted in much faster or much better solutions.

The Teamwork Method

The basis of the civilizational level, mankind has reached, is the ability of groups of human beings to work together to achieve goals a single human being is not capable to reach. For some goals self-organization of the groups is sufficient while for other goals organizational structures have been developed that are capable of making good use of the given resources, both human and material. One such structure is the project team in big companies. The main advantages of such project teams are the ability to exchange members of the team in order to find the best fitting team for a given task, the use of people with very different background and expertise, thus allowing to tackle problems that do not fit in known schemes, and the efficient but also flexible control that can be established by the team leader.

Our teamwork method reflects such a project team on the level of using only computers (or processors) as agents. Therefore we do not have to deal with emotional problems and social interactions of humans, but we have to take into account inter-processor communication and knowledge representation problems of today's computers. Our ultimate goal is a distributed system that reflects the saying "the whole is more than the sum of its parts".

The organizational structure of our teams is as follows. The team has a leader, the supervisor, to which the referees and some of the specialists report. The other specialists and all experts report to their own referee.

This is the logical structure of a team which we will use in this section. For realizing a teamwork-based system these logical components have to be assigned to the physical agents, the processors. As the problems we want to solve are problems which, at the moment, are mostly solved by very sophisticated sequential systems, we have to be very careful and efficient in our assignment of the logical components to the agents. This is because rash simulations or compromises could result in a performance of our distributed system that is inferior to that of the sequential systems. We will deal with implementational issues in the next section.

Before we describe in detail the tasks and characteristics of the components of a team, we will concentrate on the interaction between these components and on what we want to achieve by this interaction. Because we think that even on the logical level of a system restrictions to communication must play a major role, communication in a teamwork-based system only takes place during so-called team meetings. All the other time the active components, experts and specialists, work on their tasks alone, without any interaction.

In the first phase of a team meeting the referees judge the work of their experts or specialists. Their reports are sent to the supervisor as are the results of those specialists without a referee. In the second phase, the supervisor uses the reports to generate a new problem description which takes into account the new results of experts and specialists. The reports of the referees are also used to help the supervisor select the team for the next working period. The team meeting ends with the transmission of the new problem description to the selected experts and specialists.

Experts are one of the types of components that work on really solving a given search problem. Experts are search problem solvers in their own right which means that they should be able to solve several instances of a search problem theoretically alone (i.e. they should solve these instances provided there is enough time and memory available). Heuristics and strategies that are used by sequential program systems for the search problem of interest are the most likely candidates for experts. In order to avoid duplication of work, the experts used together in a team should differ in the way they construct their part of the search space (although we do not allow any arrangements between them and therefore expect that the parts overlap). This way each expert has a different view on the actual problem and can therefore produce different results.

From experts we expect, in contrast to specialists, that they can continue the work of another expert, but -as stated before- with their special view on it. This means that they all have to use the same representation for the problem description and their results. This seems to be very restrictive, but for the search problems we are interested in this is seldom a problem, because the known sequential problem solvers all have the ability to allow the user to tune them for a given problem. Therefore good compromises for the representations are known and can be used for the teamwork-based system. The use of fixed internal representations also allows the referees better access to the results of the experts.

Besides problem description and results there is other information that an expert has to store, which remains constant during a whole program run. That is, for example, the net addresses of the other processors or the initial problem description or several tables. All

this is stored in a separate memory block.

There is one other restriction on experts, namely that the single steps they do to solve a given problem are not too big, i.e. require not more than a few minutes to compute. This is necessary, because experts should stop their work for a team meeting in a stable state, i.e. after completing a defined step towards a solution. If the computation of a step of one expert takes too much time, all other experts have to wait for it, what should be avoided. Again, for the search problems we are interested in, this limitation causes no problems.

Specialists are the other type of components that work on solving a given search problem. The main difference to experts is that it is not required for a specialist to use internal representations of the problem and its work that can also be used by other specialists or the experts. This "special" representation is the reason why we call them specialists. But they must be able to communicate their results. And they must be able to translate the problem description and problem solution state used by the experts into their own representations. A typical example for a team member that is a specialist in a human team is an accountancy specialist in a team of engineers who is responsible for the detection of solution ideas that are too expensive.

Because specialists may need special data structures that are difficult and time consuming to construct, they can have a local memory -in addition to the memory block with constant data like the experts- that is not deleted as long as the specialist is member of the team (this is not the case for experts which we will see later). Due to the special data structures, specialists may also need special referees that are able to work with these structures. Specialists do not have to work on solving directly the actual problem, they can also be used to help the supervisor, for example by detecting subproblems. In a later section we will give examples for such specialists. Note that all experts can also be used as specialists. But then their results are treated differently by the supervisor (i.e. they can not be best expert, see later).

Referees have two tasks, namely the assessment of experts and specialists and the selection of good results of their experts resp. specialists. For both tasks statistical criteria can be used. How good the whole work of an expert or specialist was, can be judged by measuring the solutions and results found by it. So, an expert with many good results is better than an expert with only one good, but perhaps outstanding, result. Also the number of steps made by an expert (or specialist) and the percentage of good steps should be taken into account.

For determining the quality of a result, the referees must use knowledge about the specific area the search problem is out of. For example, in the case of an optimization problem the actual cost of a solution is a good choice. But also such solutions are interesting

that are nearly as good as the best one so far, but very different. In automated theorem proving criteria like the number of facts subsumed by a fact or the similarity of a fact to the goal to prove can be used. The more assessment knowledge an area provides the better are the judgments of the referees.

For most of the experts only the selected results find their way into the new starting point. Results that are not selected by the referees are forgotten. This avoids to blow up the search space. Only such paths that have proven to be good ones are further explored. So the work of the referees is important although they are not working on really solving the given problem.

Note that it is often necessary to use different referees for the same expert or specialist during one program run. For example, in automated theorem proving in the beginning of a run the number of facts produced by an expert is not so important than the number of good and promising facts. Later on, one goal of an expert should be to restrict the growing of the number of facts so that it has no trouble running out of memory. Using different referees takes care of this problem. So, on the logical level, a distinction between experts/specialists and referees should be made.

The *supervisor* of a team is a central control for the distributed system we propose. It has three tasks to fulfill, namely

- building a new, improved problem description as starting point of a working phase,

- planning the composition of the team for the next working phases and

- determining the duration of the next working phase.

In order to build a new starting point, which will be used as actual problem description by all experts and which is also transmitted to the specialists, the supervisor uses all the results and solutions of the expert with the best rating by its referee and adds to this set the selected results of all other experts and the specialists working on solving the problem. This total "survival of the fittest" is motivated by two reasons. First, as all experts work with a heuristic on a set of results in order to find the best result they tend to produce locally optimal results. Therefore it is a good idea to let other experts try their heuristics on the set of the best expert in order to find ways to even better results if the best expert has already reached its optimum. If it has not reached an optimum, then it should be able to continue its work which can only be guaranteed, if its set of results remains in the starting set for the next phase. The second reason that is important for search problems that include completeness properties, as for example automated theorem proving, is that such completeness properties of certain single experts can be lifted to the whole team (see [AD93]). Note that for arbitrary distributed systems that is not always the case.

The planning which experts, specialists and referees to use in the next working phase and perhaps also in further ones requires, as in the case of referees, knowledge about the area of search problems one is interested to solve. In general, we propose a frame-based representation with knowledge about special domains of the area, what experts have good success for which domain and in which combination, which expert is useful in which stages of the problem solution and much more. This knowledge, which is collected during solution attempts to many problems, represents the long-term memory of the supervisor. Its short-term memory consists of the referee reports for all experts and specialists used in the actual run. Also in the short-term memory is the information, if the actual problem can be classified into a known domain. This is provided by a specialist whose task is checking all known domains for matches with the actual problem.

Using long- and short-term memory, the supervisor is able to adapt the team to the actual problem (see later, [DK94]). The length of the next working period depends on the outcome of the planning process. If the team has not changed much, which indicates that the plan of the supervisor is successful, longer working phases can be scheduled. If most of the experts/specialists are new, shorter periods must be used in order to react fast if they do not behave as expected.

The four types of components provide a good way to incorporate the main types of knowledge that are relevant for problem solving, namely tactical control knowledge (experts and specialists), strategical control and planning knowledge (supervisor) and assessment knowledge (referees), into a distributed problem solver. It is also very important that all known sequential problem solvers for an area can easily be integrated into the system as experts or specialists.

Implementing Teamwork

If one wants to develop distribution concepts for systems that have to compete with sequential systems, it is very important to take into account the efforts for communication that can handicap a distributed system. This means that one not only has to develop a high-level distribution concept but also ideas how to implement the concept. The first question is on what platform, i.e. on what kind of machine or network, the system should run. Most concepts favor certain platforms, for example blackboard systems can be very efficient when implemented on a multiprocessor machine while the necessary shared memory requires quite some communication when implemented on a net of workstations. Teamwork was originally developed to be used on a net of workstations, but we were also able to implement the concept efficiently on a multiprocessor machine.

When it is decided which platform is used, the logical components of the concept must be assigned to the physical agents available, i.e. the processors. For ef-

ficiency's sake, it is often not recommended to assign each component to a different agent. By sharing agents interprocessor communication can be avoided and idle times of processors can be reduced. One can think of the components sharing a processor as the different *roles* an agent has to play in the system.

Teamwork allows an agent to play the roles of each logical component, i.e. supervisor, referee, expert and specialist, but only one at each time. So, at the beginning of a run one processor is assigned the role of the supervisor and given the problem description. This description is brought into the common representation used by all experts and is transmitted to all other processors after they have been assigned an expert or specialist and, if necessary, a referee. If the system operates on a net of workstations, the transmission is done using broadcast. If the system runs on a multiprocessor machine, each processor that is not acting as supervisor copies the description into its own working memory (normally a reserved part of the shared memory of all processors). Until the next team meeting the work of the supervisor is done and its processor assumes then the role of an expert or specialist.

When the end of the working phase is reached, each processor changes its role to that of a referee (except those that run specialists that do not need a referee). This way the referees can access all the data of their experts/specialists without any communication. After the refereeing process the processor which has had the role of the supervisor at the end of the last team meeting changes back into this role and receives the measurements of the experts from the referees. After the best expert is determined the processor on which this expert worked assumes the role of the supervisor while the other processor goes back to its referee role. For this change only very little information must be passed between the two involved processors.

Now, the new supervisor receives the full reports of all referees and the specialists without referee. The selected results and the data provided by the specialists can be integrated directly into the problem representation of the processor having the supervisor role. Remember that this is the representation of the best expert, the one whose whole set of results has to be part of the new starting set for the next working phase.

After determining the new team based on the referee reports, the starting set is again transmitted to all the processors. The flow of information between two team meetings is depicted in Figure 1. Dotted lines indicate communication that is virtual only, because of changing the role of a processor, while the straight lines indicate a real communication between different processors.

Because the referee reports consist only of a few results and a number indicating the measure of an expert/specialist, the main communication effort is the transmission of the set of results representing the starting point of the working phase. Both, multiprocessor

machines and nets of workstations, allow for transmitting this set to all experts and specialists simultaneously.
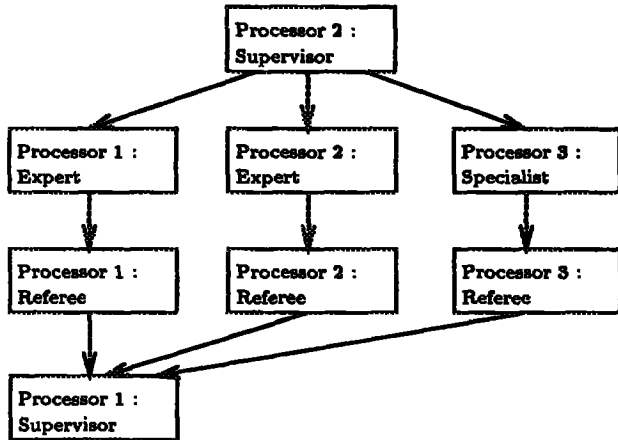

Figure 1: real and virtual data flow between two team meetings

The receiving components can use the necessity to process the results for determining the next steps they want to do. This way, even while receiving data, they can do some necessary work towards finding a solution to the actual problem. As well can specialists that use their own problem representations process the transmitted results to fit into their representations. Our experiments with teamwork-based systems (see next section) showed that the advantages of teamwork more than outweigh the efforts for repeated transmission of starting sets for new working phases.

Instantiations of Teamwork

We used teamwork to build two systems for very different search problems, equational deduction and the traveling salesman problem. In the next two subsections we will introduce these problems, sketch the experts, specialists and referees for them and summarize the results obtained with our systems.

Case Study: Equational Deduction

Automated theorem proving tries to solve the following problem:

Given: A set of axioms A and a goal G.

Question: Is G a logical consequence of A ?

In the case of equational deduction A is a set of equations $s_i = t_i$, $i = 1,...,n$ and also G is an equation $u = v$. Due to the undecidability of the problem one can only hope to get the answer YES from an automated equational prover if $u = v$ is indeed a consequence of A.

Each theorem prover is based on a set of inference rules. In the case of equational deduction these rules - forming a so-called completion procedure (see [KB70])- can be divided into two classes, generation rules and contraction rules (see [De90]). While generation inference rules deduce new equations that are added to

the set A, the contraction rules either delete elements from A or substitute elements by other, more general ones. There are always only a finite number of contractions possible until the set is in so-called normal form. The crucial steps are the generation steps. The efficiency and hence the power of a prover depend on a good selection of generation steps out of a very large number of possible steps. A proof is found, if G or a more general equation is in A. There are many selection strategies and heuristics for the generation rules, but each such strategy or heuristic is only able to solve a limited number of examples in a reasonable time.

In our teamwork-based equational prover (see [AD93] and [DF94]) the selection strategies and heuristics are used as the different experts. They use as problem representation the set of axioms A and the goal G. Note that between the team meetings each expert develops its own set A from the starting set of the last team meeting. Experts do not only use generation rules, but also try to keep their set of axioms as compact as possible by using the contraction rules and thus working on axiom sets in normal form.

The ability of an equation of A to contract other equations or even the goal is a major criterion used by the referees for both determining the success of an expert and selecting good results. The more equations of A could be eliminated or changed by an equation the better it is. The size of A, the number of steps made since the last team meeting, the number of potential generation steps and several other criteria are also used to get a statistical measure of the success of an expert. The number of potential generation steps with one equation, the type of this equation, the similarity of it to the goal and other measures can be used to select good single equations for sending to the supervisor. In order to limit the amount of communication the referees are only allowed to report up to a given number of equations to the supervisor. This number was 5 for the examples given below.

Due to lack of space and because we used fixed teams for each example of Table 1, we will not go into detail how the supervisor plans the composition of teams. More about this can be found in [DK94]. But there are two important results we want to mention. Firstly, it is possible, by using reactive planning, to adapt a team to a given problem. We were able to solve all examples of Table 1 and many more starting with the same team and exchanging experts according to knowledge about domains of interest and good suited experts for them (as long-term memory) and according to the measures provided by the referees (as short-term memory). Because planning and trying out experts takes some time, the run-times are slower than those of the best known teams of Table 1 for the examples, but we were able to solve them in acceptable time.

Secondly, a very important tool for planning was a specialist that assisted the supervisor. This specialist tried to detect subsets of equations in A that define a

known domain. The results of this specialist allowed the supervisor to know which knowledge of the long-term memory it should use.

Example	Team	Best Expert
p2.a	5.41	79.52
p2.b	5.38	—
p8.b	56.84	—
p9.a	8.66	19.57
p9.b	8.44	50.95
p10	23.20	—
bool5b	72.86	—
ra2	125.37	227.88
sa2	10.75	—
herky3	6.81	16.09
luka3	81.68	—
ring	307.96	3019.00

Table 1: run-time comparison team with best sequential runs (in seconds, real time)

Let us now take a closer look at the results we achieved using teamwork. The examples of Table 1 are collected from [AD93] and [DF94] where one can find the descriptions of the examples. The teams always consisted of two experts and the runs were made on a net of 2 SUN-ELC workstations. In order to allow a judgment of the results the row entitled "best expert" contains the run-times of the best of our experts for the example working alone. This is the best known result one can obtain in our system if one does not want to do a distributed search for a proof.

Note that some of our experts working alone are quite comparable to the best known proof systems, as for example the Otter-system ([Mc94]) (see also [DF94]). Therefore the speed-ups obtained by using teamwork are quite remarkable. Half of the examples could not be solved sequentially in reasonable time (i.e. under 3 hours run-time). An analysis of the effects responsible for these speed-ups, namely one expert taking over all results of another one and working in a better direction, but not being able to generate all results alone, and secondly inferior experts and specialists providing missing pieces of a solution as selected results, can be found in [DS94]. So, teamwork has proven that it is able to enhance the abilities of sequential provers.

Case Study: Traveling Salesman

The traveling salesman problem (TSP) is often called the standard example of an NP-complete optimization problem ([LLRS85]).

Given: An $n \times n$ cost matrix $C = (c_{ij})$, $i, j \in \{1, ..., n\}$.

Question: What is a permutation π of $\{1, ..., n\}$, such that $\sum_{i=1}^{n} c_{\pi(i), \pi(i \bmod n + 1)}$ is minimal?

Even if we assume that all c_{ij} are positive integers, the matrix is symmetric (i.e. $c_{ij} = c_{ji}$ for all i,j) and the triangle inequality holds (i.e. $c_{ik} + c_{kj} \leq c_{ij}$) the problem remains hard, i.e. NP-complete (see [AHU74]). Therefore many authors concentrated on finding sub-optimal algorithms for solving this problem that have an acceptable (i.e. polynomial) run time (for example

tour construction heuristics like Farthest Insertion or Cheapest Insertion [RSL74], tour improvement heuristics like 3-opt [Li65] or Lin-Kernighan [LK73], or genetic algorithms as described in [Mi92]).

These approximative solution methods are the basis for our teamwork approach to solving the TSP. They are used both as experts and as specialists. But also optimal solution methods, for example branch and bound based ones (see [LLRS85] for several examples) can be incorporated as specialists into a teamwork-based distributed system for solving the TSP.

The problem representation used by the experts consists of the cost matrix, a list for each i with the c_{ij} sorted in ascending order (see later) and a list of solutions (tours) to the TSP sorted in descending order by tour length. While the cost matrix is part of the permanent data of the experts (and the specialists), the solutions represent the results and can be exchanged with the list of the winner of a working period during a team meeting. Since it is rather easy to obtain some -often not very good- solutions the main part of the work of the team is the improvement of solutions. Therefore we use tour improvement methods as experts. There are several tasks that prepare or help finding solutions. Such tasks are, for example, sorting the rows of the cost matrix, the initial construction of several tours or the calculation of lower bounds for the optimal tour length. These tasks are dealt with by specialists in the first few working periods.

Since TSP is an optimization problem it is quite easy to find criteria that can be used by the referees. For judging an expert its shortest solution, the fact whether this solution is new or not, the number of new solutions found, the best m solutions and the differences between those solutions should be taken into account. The similarity of two tours can be indicated by counting the number of edges that are in both tours. Then the difference is n minus the similarity.

The referees determine the solutions they send to the supervisor by choosing the shortest solution of the expert and other good tours (i.e. tours whose cost is within a certain percentage of the best) that are very different from the best one. We use difference of solutions as criterion, because we want to avoid sending the whole team into a local minimum.

The planning task of the supervisor is easier than in the case of theorem proving, because it is easy to find several stages for constructing a near-optimal solution. In the first stage, specialists should be used to construct several solutions and to prepare the further work by sorting the rows of the cost matrix or by determining whether the actual problem is in a known special domain. Then, in the middle stage, experts should be used to improve the best solution found so far. Again, experts that do not contribute to better solutions can be exchanged. Finally, when there are only slight improvements from team meeting to team meeting experts and specialists should be used that try

to leave local minima, for example a genetic algorithm with a high mutation rate.

Exp.	best sol.	best t. sol.	best exp. sol.	comp. team sol.
kroB100	22141	22141	22251	22220
gr202	40160	40160	40777	40672
pcb442	50778	50931	52393	51941
att532	27686	27775	28198	28196
gr666	294358	296704	303744	302113

Table 2: comparison best solutions of team and single runs of experts

Exp.	best t. sol.	best exp. sol.	comp. team sol.
kroB100	9.87	145.43	5.36
gr202	43.58	87.27	18.96
pcb442	538.39	696.71	58.00
att532	1449.28	318.77	110.94
gr666	1552.94	4728.10	140.18

Table 3: comparison time (in sec) for finding solutions of team and single runs of experts

In our first prototype we used only two specialists, the sorting specialist for the rows of the matrix and a tour construction specialist using Farthest Insertion and two experts, 3-opt and a very parameterized version of a genetic algorithm described in [Mi92]. Our first results can be examined in Tables 2 and 3. The numbers in the names indicate the number of cities of the example. The examples and the best known solutions are taken from the TSPLIB of [Re91]. The results were obtained on a cluster of 2 Sun-Sparc 10 machines and the times state when the solution first occured.

If we take a look at the results of the tables, we can first observe that our teams are able to find better solutions than the experts working alone, thus again showing a synergetic gain by using teamwork. While our teams are always within 1% of the best known solution, the best experts come only to solutions from between 1% to 3.2% of the best known one.

Our teams may need more time for finding such good solutions (as for example att532). In order to demonstrate the speed improvement, we give in the last column of Table 3 the run times (and in Table 2 the solutions) of our teams until a solution that is comparable to the best found by an expert is found. Even if we multiply these times by 2, thus taking into account that we use 2 processors instead of one, the team is much faster than the expert alone. Again, the cooperation and competition of heuristics proves to be successful.

Related Work

This paper addresses three research areas, namely distributed search, automated theorem proving and the traveling salesman problem. In all these areas much research has been done that can be related to our work. In the following we will only discuss such works that are closely related to teamwork.

In the area of distributed search the main part of work focused on the distribution of special search processes, as for example theorem proving. Only in the last few years there were attempts to develop more general approaches to distributed search. For example, the works of Lander and Lesser (see [LL92], [LL93]) deal with the class of search problems that have the goal to reach a consensus or, at least, a compromise between different needs. Such problems can be distributed naturally by simply assigning one agent to each opinion or need. The problems in this class are the construction of a framework for supporting information exchange between very independent agents, the development of conflict solution mechanisms and the reuse of agents.

In the areas of our two case studies there has also been done some work regarding distribution. In the area automated theorem proving there have been several attempts to parallelize and distribute the search for a proof. Although many approaches were only concerned with parallelizing a sequential algorithm (for example ROO, see [LM92]), there are approaches that focused on a distributed search with totally different behaviour than sequential approaches (for example the DARES system, where the input facts were distributed among the agents and cooperation had to take place in order to solve an example, see [CMM90]). An overview of parallel and distributed theorem proving can be found in [SS94].

In the area of solving optimization problems the advantages of cooperation between different approximation methods have been observed very early, leading to the design of composite (sequential) methods (see [GBDS80], [Jo90]). But only recently work has been done to actually distribute the search process, namely the A-teams of [ST93]. The members of an A-team are, as the experts in our teamwork approach, approximation algorithms. They communicate by putting all their results into several memory regions which can be read by all members, thus allowing an Asynchronous organization. Like teamwork, the A-team approach allows for synergetic effects, but, since this approach lacks referees and a supervisor, there is no possibility to automatically adapt the team to the actual problem. Also there is much more communication involved than in the case of our teamwork approach. This could cause problems when using a net of workstations (it would definitely cause problems if one wanted to use A-teams for automated theorem proving).

Conclusion

With teamwork we presented a general method to distribute search problems that cannot be partitioned into subproblems (or subtasks) in a natural manner. Such problems are, up to now, solved by sequential systems that use various heuristics to guide the search process. Teamwork enforces competition and cooperation between such heuristics and we showed the synergetic effects it allows with two case studies in automated

theorem proving and the solution of optimization problems. In both case studies, the distributed system using teamwork generated faster or better results than each of the sequential heuristics when used alone.

By providing a concept that restricts communication very much, but allows an efficient implementation, teamwork can be used on networks of computers without any special hardware. It can also be used on multiprocessor machines.

The components of teamwork-based systems - experts, specialists, referees and the supervisor- allow the adequate use of control knowledge (both tactical and strategical) and assessment knowledge. Not only all the heuristics and tricks of the known sequential systems for a search problem can be taken over, but also very specialized pieces of knowledge that contradict other pieces or can only be applied very seldom can be integrated into a teamwork system without loss of efficiency. In contrast, in those cases this knowledge is applicable it is used in a very efficient way. This is due to the ability of the system to adapt to a given problem.

Further research regarding teamwork shall be directed into the use of a hierarchy of teams and the use of teamwork for other instances of the class of problems it was designed for, as for example scheduling problems.

References

[AD93] Avenhaus, J. ; Denzinger, J.: Distributing equational theorem proving, In Proc. 5th RTA, Montreal, LNCS 690, 1993, pp. 62-76.

[AHU74] Aho, A.V. ; Hopcroft, J.E. ; Ullmann, J.D.: *The Design and Analysis of Computer Algorithms*, Addison Wesley, 1974.

[CMM90] Conry, S.E. ; MacIntosh, D.J. ; Meyer, R.A.: DARES: A Distributed Automated REasoning System, In Proc. AAAI-90, 1990, pp. 78-85.

[De90] Dershowitz, N.: A Maximal-Literal Unit Strategy for Horn Clauses, In Proc. 2nd CTRS, Montreal, LNCS 516, 1990, pp.14-25.

[De91] Denzinger, J.: Distributed Knowledge-based Deduction using the Team Work Method, SEKI-Report SR-91-12, University of Kaiserslautern, 1991.

[DF94] Denzinger, J. ; Fuchs, M.: Goal oriented equational theorem proving using teamwork, In Proc. 18th KI-94, Saarbrücken, LNAI 861, 1994, pp. 343-354.

[DK94] Denzinger, J. ; Kronenburg, M.: Planning for distributed theorem proving: The team work approach, SEKI-Report SR-94-09, University of Kaiserslautern, 1994.

[DS94] Denzinger, J. ; Schulz, S.: Recording, Analyzing and Presenting Distributed Deduction Processes, In Proc. PASCO'94, Linz, 1994, pp. 114-123.

[GBDS80] Golden, B.L. ; Bodin, L.D. ; Doyle, T. ; Steward Jr., W.: Approximate traveling salesman algorithms, *Operations Research* 28, 1980, pp. 694-711.

[Jo90] Johnson, D.S.: Local Optimization and The Traveling Salesman Problem, In Proc. ALP-90, LNCS 443, 1990, pp.446-461.

[KB70] Knuth, D.E. ; Bendix, P.B.: Simple Word Problems in Universal Algebra, *Computational Algebra*, J. Leech, Pergamon Press, 1970, pp. 263-297.

[Li65] Lin, S.: Computer Solutions of the Traveling Salesman Problem, Bell Syst. Tech. J. 44, 1965, pp. 2245-2269.

[LK73] Lin, S. ; Kernighan, B.: An Effective Heuristic Algorithm for the Traveling Salesman Problem, *Operations Research* 21, 1973, pp. 972-989.

[LL92] Lander, S.E. ; Lesser, V.R.: Customizing Distributed Search Among Agents with Heterogeneous Knowledge, In Proc. 1st International Conference on Information and Knowledge Management, Baltimore, 1992, pp. 335-344.

[LL93] Lander, S.E. ; Lesser, V.R.: Understanding the role of negotiation in distributed search among heterogeneous agents, In Proc. IJCAI-93, Chambery, 1993, pp. 438-444.

[LLRS85] Lawler, E.L. ; Lenstra, J.K. ; Rinnooy Kan, A.H.G. ; Shmoys, D.B.(eds): *The Traveling Salesman Problem*, John Wiley and Sons Ltd., 1985.

[LM92] Lusk, E.L. ; McCune, W.W.: Experiments with ROO: a Parallel Automated Deduction System, in Fronhöfer, Wrightson (eds.), Parallelization in Inference Systems, LNAI 590, 1992, pp. 139-162.

[Mc94] McCune, W.W.: OTTER 3.0 Reference manual and Guide, Tech. Rep. ANL-94/6, Argonne National Laboratory, 1994.

[Mi92] Michalewicz, Z.: *Genetic Algorithms + Data Structures = Evolution Programs*, Springer Artificial Intelligence, 1992.

[Re91] Reinelt, G.: TSPLIB - A Traveling Salesman Problem Library, *ORSA Journal on Computing*, Vol.3, No.4, 1991, pp. 376-384.

[RSL74] Rosenkrantz, D. ; Stearns, R. ; Lewis, P.: Approximate Algorithms for the Traveling Salesperson Problem, In Proc. 15th IEEE Symposium of Switching and Automata Theory, 1974, pp.33-42.

[SS94] Suttner, C.B. ; Schumann, J.: Parallel Automated Theorem Proving, in Kanal, Kumar, Kitano, Suttner (eds.) Parallel Processing for Artificial Intelligence, Elsevier, 1994.

[ST93] de Souza, P.S. ; Talukdar, S.N.: Asynchronous Organizations for Multi-Algorithm Problems, In Proc. ACM Symposium of Applied Computing, Indianapolis, 1993, pp. 286-294.

[TSM93] Talukdar, S.N. ; de Souza, P.S. ; Seshashayee, M.: Organizations for Computer-based Agents, *Journal of Engineering Intelligent Systems*, 1993.