

# On Cooperation between Evolutionary Algorithms and other Search Paradigms

**Jörg Denzinger**

Fachbereich Informatik, Universität Kaiserslautern  
Postfach 3049, 67653 Kaiserslautern, Germany  
denzinge@informatik.uni-kl.de

**Tim Offermann**

Fachbereich Informatik, Universität Kaiserslautern  
Postfach 3049, 67653 Kaiserslautern, Germany  
offerman@informatik.uni-kl.de

**Abstract-** We present a multi-agent based approach for achieving cooperation between search systems employing different search paradigms. The search agents periodically interrupt their search, select interesting information from their states that is transmitted to the other agents, filter the information sent to them with respect to their own demands, integrate the remaining information into their search, and then continue the search. There are different kinds of information to be exchanged and the selection is both success- and demand-driven.

We demonstrate the usefulness of this approach by coupling a search system based on a Genetic Algorithm and a branch-and-bound based system for job-shop-scheduling. Our experiments show that the cooperation results in finding better solutions within a given time limit and in finding solutions comparable to those generated by the best system working alone in less time. The speed-up factors for some examples even exceed the number of agents (computers) used.

## 1 Introduction

Search is a central problem solving technique employed by many programs. For most problem areas there exist systems based on different search paradigms. For example, for discrete optimization problems early systems employed either enumeration techniques, local optimization techniques or techniques that try to intelligently partition and restrict the solution space, like branch-and-bound. In the last years, also evolutionary approaches have shown quite some success as a search paradigm.

All these paradigms have different strengths and weaknesses. For example, usually evolutionary approaches cannot guarantee an optimal solution but deliver a good solution quite fast. A branch-and-bound based approach usually guarantees to find an optimal solution, but takes quite some time to come up with some very good solutions. Since it is very difficult to decide a priori what paradigm to employ in order to solve a given instance of a search problem, people from several application areas have suggested to find ways to combine the different paradigms (see, for example, the challenges in [Ka97] or [SKM97]).

In this paper we propose to combine the different paradigms by letting search systems, each using one of the paradigms, cooperate as agents of a multi-agent system. Each agent gets the whole instance of the problem to solve, tries to

solve the instance according to its search paradigm, and periodically interrupts its search in order to send information to other agents and to receive information from them. Information to exchange can be positive, i.e. information intended to direct the search process towards the search goal, and negative, i.e. information intended to avoid possible steps in the search process. The exchanged information can directly be included in the search state of an agent or it can influence the control the agent employs in its search.

Since each action taken by an agent during its search results in possible information to exchange, it is very important to limit the information that is really exchanged. In our concept this is the task of so-called referees. Send-referees select information of all types on the side of the sending agent, mostly concentrating on the success the agent achieved with a piece of information or using some ideas of the general needs of other agents. Receive-referees provide an additional filter on the side of the receiving agent, using criteria that measure the current and probable future needs of their agents. Our approach has several advantages among which are the easy integration of existing search systems due to the open character of the multi-agent system and the synergetical improvements both in quality of solutions and speed in finding them.

In the area of discrete optimization problems, Genetic Algorithms (GA), as one representative of the evolutionary algorithms paradigm, and Branch-and-Bound (B&B) algorithms, as a representative of the and-tree search paradigm, result in very different systems that are very good candidates for a fruitful cooperation. In this paper we demonstrate the success of our approach by using it for the cooperation of such systems for the job-shop-scheduling problem. Our experiments show that our multi-agent system indeed finds in a given time period better schedules than each of the used systems alone (if possible) and finds schedules of a quality comparable to that of the best system used in less time. In many cases, the speed-up factor even exceeds the number of agents (and so of computers) used.

## 2 Cooperative Heterogeneous Search with TECHS

In order to present our TECHS approach (TEams for Cooperative Heterogeneous Search) we first have to briefly define what search agents and what possible types of information to exchange are. A search agent is a search process with additional communication abilities. A search process

uses a search model and defines a search control. A search model consists of a set of possible search states and a transition relation that defines the possible successor states to each state. Since usually there are several possible successor states to a search state, we need a search control that resolves this indeterminism by selecting for a search state exactly one of the possible successor states defined by the transition relation.

Different search paradigms use different structures to represent search states. For example, a B&B search process has a search model where the set of search states consists of and-trees, while the search model to a GA (or an evolutionary algorithm in general) uses a set representation, namely a subset of the set of possible individuals, for search states. Information that may help a search agent can either be integrated in the search state or it may only be used by the search control without integration in the state (for example as parameters of the control). The search state usually is intended to represent *positive information*, that is information that helps finding a solution to a given instance of a search problem. Some search models allow to represent *negative information*, that is information that does not lead towards finding a solution, but characterizes attribute values that are definitely not part of a solution. Also information that is only used by the search control can either be positive or negative.

Our TECHS approach aims at providing a framework for the cooperation of search systems using different search models. This cooperation is based on the exchange of information of all types mentioned above, provided that a search system can make use of the different types. In order to not overwhelm the systems by too much data, information of all types is filtered.

In a TECHS-based system, search processes are provided by the different search systems. By adding so-called send- and receive-referees and communication channels between the processes, we get search agents. In an actual team, there may be several agents that are based on the same search system, but then they have to use different parameter settings so that they produce different sequences of search states.

The task of a *send-referee* is to select, out of the current search state (and perhaps earlier states) of its agent, information of all possible types that should be communicated to the other agents. Although the criteria used by send-referees have to depend on the search problem and the search agents to be used, we can characterize some general principles. In order to select positive information to be integrated into search states of other agents, send-referees should use a success-driven approach based on the history the information had with respect to the search of an agent. Parts of search states that enabled good transitions should be preferred to parts that only enabled bad ones. Negative information that should be integrated into the search states of other agents can describe unsolvable sub-problems, for which the proof that they are really unsolvable took some time.

In order to select information that can be used by the controls of other agents, a send-referee needs some general in-

formation about these agents. Such demand-driven criteria involve rating the information to select with respect to the control strategy of other agents (if they are of the same type as the sender) or with respect to certain attributes solutions should have or not have. In addition to both types of criteria a further criterion should be that it is very unlikely that the other agent will find the information on its own.

In order to communicate information of all types *data formats* are needed to represent the information. Usually, there are certain formats used by humans to describe the search problem and possible solutions (although the search systems internally might use totally different data structures) that can be used for communication by all systems. An additional task of send-referees is to transfer the internal format of an agent into the communication format.

While send-referees already drastically restrict the amount of information send to other agents, the task of the *receive-referees* is to filter out all information that does not meet the demands of their particular search agent. Criteria for all types of information try to measure the impact of the information on the further search of the agent. This is especially important for positive information, since it tends to broaden the search space if it is not useful for the problem at hand. Filtering negative information is not so critical, but if an agent accumulates too much negative information then the selection process for the next search steps can become time consuming. Receive-referees also have to transfer information given in the communication format into the format of their agents.

Each agent in a TECHS-based system repeats a cycle consisting of a working phase in which the agent concentrates on its search and a cooperation phase in which its send-referees (there can be a send-referee for each of the other agents but a send-referee might also select information for several agents) selects information to be sent away and its receive-referee (there is always only one) filters the information received from the other agents. The information selected by the receive-referee is then integrated in the search state and the search control and a new cycle begins. If all agents use the same time interval for working phases one can organize the information interchange in a synchronized manner, else it has to be asynchronous.

Thus, when building a TECHS-based search team, the following problems have to be solved:

- data formats for the different types of information must be found,
- the used search systems must be modified to provide send-referees with information,
- the search systems must also be modified to enable the integration of the information selected by their receive-referees into search state and search control,
- send- and receive-referees must be developed.

While the first two problems are typically easy to solve, since most search systems already provide their users with much in-

formation, the third problem can be very hard (see Section 4). One can start with rather primitive send- and receive-referees and can refine them later on, so that the fourth problem is not so hard, again. TECHS has already been successfully used to let several (already existing) theorem provers cooperate, resulting in synergetical speed-ups (see [DF99] and [DD98]).

### 3 The Job-Shop Scheduling Problem

Job-shop scheduling (JSS) problems are among the more difficult problems of the class of NP-complete problems. They are of quite some practical interest, too, since many companies need to solve JSS problems very often in their production processes. In a JSS problem the task is to process  $n$  jobs on  $m$  machines. Each job consists of a sequence of operations, each operation has to be performed by a different machine. Each machine can only do one operation at a time and a started operation cannot be interrupted. A solution to this problem consists of a schedule of the operations necessary for the jobs on each machine, fulfilling certain requirements and being optimal with respect to a certain quality measure. Usually, the requirements only are that the operations for each job must be performed in a given order and the quality of a schedule is the time needed until the last operation in the schedules is performed. For some variants of job-shop scheduling there are more requirements and also other notions of quality possible, but we will use in this paper the basic variant described above.

#### 3.1 The JOB-SHOP system

The JOB-SHOP system (see [BJS94]) is a B&B system for solving the JSS problem, implemented in C. B&B based search uses (in theory) and-trees to represent search states and transitions either close leaves in such a tree, if they represent a solution or have a bound that is not better than the best currently known solution, or expand one leaf of the current tree by adding new leaves for each of the different values some problem specific variables can have. Which leaf to expand and what variable(s) to use is determined by the search control. The search stops with a state in which all leaves are closed.

There are several different B&B algorithms to solve the JSS problem. We have chosen JOB-SHOP because it was available and the reported results were not bad. Each node of a state in JOB-SHOP contains a graph-model view (see [RS64]) on the problem instance. The nodes of such a graph contain the operations. The operations that have to be performed by the same machine are connected by disjunctive arcs forming a complete subgraph. On the other hand all operations of a given job are connected by directed arcs in the order of execution. An ordering of the operations of a given machine is established by fixing the direction of the disjunctive arcs. JOB-SHOP fixes several arcs when extending a leaf of the search state using the method of Carlier and Pinson (see [CP90]).

JOB-SHOP computes for each leaf of a state a bound that approximates the quality of the best solution contained in the set of solutions represented by the graph in the leaf (the graphs are an alternative format for partial schedules, see Section 4). As already stated, this bound is used to close leaves. Typically, in a B&B system the bound is also used to select the next leaf to expand, but in JOB-SHOP a depth-first approach for controlling the search has been implemented instead. Only locally, i.e. among the successors of a node, a selection criterion involving the bound is used.

#### 3.2 Our GA system: NY

As in the case of B&B search systems, there are also many different approaches to build systems based on GAs to solve the JSS problem. A search agent employing a GA uses sets of complete solutions (individuals) as search states and a transition is described by (a sequence of) genetic operators (including an operator for deleting one or several individuals from a set). The search control of the agent has to select the genetic operators and the individuals (parents) for their premises from the current search state. Usually, this involves both random factors and the fitness-values of the individuals of the current state.

For our experiments, we used a GA based on the idea of Nakano and Yamada (see [NY92]) which employs a special version of the Giffler-Thompson algorithm (see [GT60]) both to create schedules and in the crossover and mutation operators. The individuals of a state describe full schedules. The used genom consists of a number string recording for each operation the starting time. The correctness of all schedules is guaranteed by using the Giffler-Thompson algorithm in each genetic operation.

For the creation of a schedule, the Giffler-Thompson algorithm is guided by a random selection process for the operations. For a crossover, the construction of the new schedule is guided by the parents. First, the selection process of the algorithm is guided by the starting times of one parent and at a randomly selected crossover point the other parent is used as guidance. Finally, the mutation operator tries to keep the operation ordering for all but one machine as in the parent. For this selected machine, a random priority string guides the Giffler-Thompson algorithm. The resulting schedule for the selected machine then might necessitate adjustments in the schedules of the other machines.

The search control we use implements a steady-state algorithm using an exponential ranking selection for the parents for a genetic operator. The newly generated individual replaces an individual that is selected by an inverse exponential ranking (worst individuals get highest probabilities) and that is not the best individual of the current state.

### 4 TECHS for Job-Shop Scheduling

As stated in Section 2, the first task when building a TECHS-based system for an application domain is to find appropriate

data formats for the different types of information. For our TECHS-based system to solve JSS problems, the central data format that we will use for all types of information is the *partial schedule*.

In contrast to a full schedule that contains for each operation of each job the exact start time and that fulfills all requirements on the problem, a partial schedule only consists of a set of ordering conditions that are based on two relations between operations on a machine. The strong ordering relation  $op_1 \triangleright op_2$  indicates, that the operation  $op_2$  directly follows operation  $op_1$ , which means that no other operation is performed between them on the machine (but a delay between the end of operation  $op_1$  and the start of  $op_2$  is possible). The weaker relation  $op_1 > op_2$  indicates, that operation  $op_1$  is performed by the machine before  $op_2$  is performed, but there may be other operations that can be performed after  $op_1$  and before  $op_2$ . The relations  $\triangleright$  and  $>$  are only defined for operations on the same machine.

A set of ordering conditions is a partial schedule if there is a full schedule that fulfills all these conditions. If each ordering condition of one partial schedule is also a condition of another partial schedule (or consequence of the conditions of this other schedule), then the first schedule *matches* the second one.

It is possible to use partial schedules as data format for all types of information, because full schedules are a special case of partial schedules (if a partial schedule can be extended to exactly one full schedule, assuming that each operation on each machine is started as soon as possible, we then identify partial and full schedule) and partial schedules can be interpreted very differently. For example, a partial schedule interpreted as positive information to be included into a search state defines a set of full schedules that can be generated out of it (and that include all the conditions already defined by the partial schedule). Interpreted as negative information a partial schedule indicates that all full schedules that are matched by it are not optimal.

#### 4.1 The necessary modifications on the systems

Although TECHS is intended to be used to let already existing search systems cooperate, we had to implement NY from scratch. Naturally, we included everything necessary for applying TECHS so that no modifications should have been necessary. Nevertheless, early experiments showed that the integration of schedules obtained from the B&B agents tended to lead to a premature convergence. By already choosing a selection mechanism based on ranking of individuals instead of mechanisms like the wheel of fortune that tend to favor individuals with very high fitness too much, we thought that such problems would not occur.

But they did occur and therefore we had to modify the search control in the following way: We added an extra value to each individual that measures the “alienness” of it, i.e. how many transitions the individual is away from an ancestor that was injected into the search by a B&B agent. To make

things easier, we defined an `alien-max-value` that was assigned to all individuals integrated by the receive-referee. When generating a new individual by mutation or crossover, this individual gets the alienness value of the parents (the maximum thereof) minus one, or it obtains zero if the parents already have zero as value. All individuals of the initial population also have an alienness of zero. Genetic operations with parents whose summed up alienness is greater than `alien-max-value` are forbidden. This way, “incest” between injected individuals and/or their descendants is prohibited until they are “unrelated” enough. Note that without injected individuals we have a “normal” GA behavior.

The modifications on JOB-SHOP were bigger than we thought at first. The problem with JOB-SHOP was that only depth-first search of the state tree was intended by its developers. This influenced several places of the code. For example, the necessary backtracking was managed with the help of global variables. Since we wanted a flexible control allowing for different control strategies and the use of control information provided by other agents, we had to redesign parts of the system.

Now, JOB-SHOP really represents the search state as an and-tree (and not as a path with backtracking points) and additionally the search control can use a list of open leaves of the tree. This guarantees that the send-referees can easily access all necessary information and also different leaf selection strategies can be used. We added to the depth-first strategy a (global) best-first and a breadth-first strategy. Unfortunately, this redesign resulted in a performance loss of a factor of two when comparing the original system with our modified one using depth-first. But the new strategies are often able to compensate this loss for needing fewer transitions until a solution is found than depth-first.

#### 4.2 The send-referees

Send-referees select information of all types out of the current search state and the search sequence produced so far (provided that a certain information type is defined for the agent). While the send-referees mainly base their selection on the success the pieces of information produced for the agent, it is also possible to include some general knowledge about the demands of the receiving agents in the selection process. Because the usage the receiving agents make of a certain information determines its type, we will present in the following send-referees for each type of search agent and give the selection criteria we used for the different types of information.

##### B&B → B&B

B&B agents exchange positive and negative information to be integrated into the search state. So far, no control information is exchanged. The only positive information that such an agent can integrate into its search state is a full schedule that is better than the best full schedule known to the agent. Therefore the send-referee always selects the best known full

schedule, if there has been some change since the last cooperation phase. Otherwise, no positive information is selected.

Negative information to be exchanged are partial schedules describing closed subtrees of the current state. The more nodes this subtree has, the more work was necessary to get this information and the more important it is to communicate this information. Also the subtrees should be bigger than a certain minimal size, so that a substantial gain can be expected for other agents. In our system the `n-select-size` partial schedules representing the biggest closed subtrees with more than `min-size` nodes are selected.

### **B&B → NY**

Agents using a GA can, so far, only use positive information to be included into the search state. Our send-referee selects the partial schedules describing the paths in the current search state of its agent that have the best bound-values (and have not been communicated already). The `p-select-size` best partial schedules with respect to this criterion are selected.

### **NY → B&B**

Agents using a GA can only produce positive information. B&B agents can use such positive information both as control information and as information to be integrated into the search state. As in the case of the B&B → B&B send-referee, the positive information to be integrated into the search state is the best individual of the current state of the NY agent.

In order to find criteria for the selection of positive control information, we have to explain first how a B&B agent can use such information. The selection of the next leaf to expand for a B&B search system is not easy. Even when using a best-first strategy many leaves have the same bound, so that additional criteria like FIFO (first in, first out) that include no problem-specific knowledge have to be used. This results in many transitions that would never be performed if other leaves were selected earlier.

It is possible to use full schedules to guide the search control of B&B agents. If there is a path in the current state of the agent that is represented by a partial schedule that matches the full schedule, then the advice from the NY agent can be interpreted as a suggestion to expand the leaf of this path. This can be repeated for several (`contr-count`) transitions or one could simply switch to a depth-first control for these transitions. In both cases, the agent is directed into other areas of its and-tree and if the search control includes a criterion that prefers leaves that are deeper in the tree, then the result can be that the “normal” search control performs more transitions in this other area. We have chosen the second alternative.

Hence, candidates for individuals to select by the send-referee are individuals representing very good solutions. In addition, they should represent different areas of the search tree of the receiving agent than its currently focused area. Since the send-referee does not know what this current area is, an additional criterion should be that the individuals are

very different (this already proved to be very successful for TEAMWORK, see [DS97]). The best `ind-select` individuals with respect to these criteria are selected.

### **NY → NY**

Although we will not use this send-referee in the experiments, we have implemented it. The only type of information exchanged between NY agents are positive individuals to be integrated into the current search state, i.e. the well-known migration of individuals. Useful criteria for the selection are quality of the solutions and their difference, again.

## **4.3 The receive-referees**

Receive-referees filter incoming information in order to select information that meet the current needs of their search agents. In the following, we will describe the criteria used by the two receive-referees for the two types of agents and we will also sketch how the selected information is integrated into the search process of the agents.

### **NY**

A NY agent only receives partial schedules to be integrated into its search state. Since the individuals are full schedules, the receive-referee has to extend partial schedules to full ones (if they are not already full ones). This can easily be done. The criteria used for selection resemble the criteria used by send-referees: quality and difference. But now the difference is measured with respect to the current best individual of the search agent. The `ind-count` best individuals are selected and integrated into the search state as if they were generated by a genetic operator.

### **B&B**

The receive-referee of a B&B agent has to filter positive and negative information to be integrated into the search state and positive control information. From the positive full schedules representing the information to be integrated into the search state only one, namely the one with the highest quality, is selected, provided that this quality is higher than the quality of the best known solution so far. If such a schedule is received, then it is integrated by checking for all open leaves if their bound is worse than this new solution. If this is the case, then the leaf is closed.

The partial schedules representing negative information are used by the agent as an additional criterion to close a leaf. If such a negative partial schedule matches the partial schedule represented by the leaf, then it can be closed. As a matter of fact, this criterion can also be used to close an inner node of the search tree. Not every negative partial schedule can be used immediately. Therefore they are stored by the agent in an additional list that is used to check newly generated leaves.

The receive-referee selects all negative schedules that are not matched by a negative schedule in this additional list. It

also checks if there are schedules in the list that are matched by new ones. If this is the case, then they are removed from the list.

The selection of the schedules representing control information is performed in two steps. Firstly, all received schedules that represent closed paths in the current tree are filtered out. Secondly, all schedules that represent the area of the search tree that is currently investigated by the agent are removed. From the remaining schedules the `pos-contr-count` best are selected and they are used by the search control for determining the next transitions as described earlier. It should be noted that `pos-contr-count*contr-count` should not be so large that the agent has no chance to perform other transitions before the next cooperation phase.

## 5 Experiments

In order to evaluate our TECHS approach we conducted several experiments using examples stemming from the OR-library (see URL: <http://mscmga.ms.ic.ac.uk/info.html>, keywords: scheduling/job-shop). We use the names reported there and also the information on the best known solutions. We selected only examples that are hard to solve for JOB-SHOP, which means that they require more than 25000 seconds run-time or could not be solved at all. Naturally, the effects and improvements by cooperation can be examined best on such hard examples.

We have a cluster of 3 Sun Sparc 10 machines which limits our experiments to teams involving only 3 search agents. In preliminary experiments our expectation was verified that the best teams consist of two incarnations of JOB-SHOP and one incarnation of NY. We expected this because JOB-SHOP needs much more time for performing a transition than NY.

Since we have 3 control strategies for our B&B agents, there have been 3 different teams. Due to lack of space we report in the tables only the results of the best team and compare it to the results of its team members when working alone. For each experiment, we allowed for a maximal run-time of 28800 seconds (i.e. 8 hours) real-time. The parameters of Section 4 were set as follows: `alien-max-value` = 10, `n-select-size` = `p-select-size` = 3, `min-size` = 300, `contr-count` = 50, `ind-select` = 5, `ind-count` = 3, and `pos-contr-count` = 3. The agents had working phases of 125 seconds and the communication was synchronized using the functions of the UNIX socket concept. The NY agent used a population size of 2600 and, due to the long run-time allotment, a mutation percentage of 50.

Table 1 shows that our TECHS team produced better schedules than all its agents working alone (except for example la26, for which the team and one single agent found the optimal solution). This means that the cooperation by TECHS produces better results than a pure competition approach that runs several agents in parallel and returns the result of the best one. Note that for examples la26, la36, la37, and la39 the TECHS team was able to produce the optimal

Ex.	NY	B&B <sub>1</sub>	B&B <sub>2</sub>	TECHS	Opt.
la26	1306	1254	1218	1218	1218
la27	1300	1330	1298	1271	1235
la28	1300	1333	1329	1251	1216
la29	1249	1241	1258	1228	≥1142
la36	1323	1304	1291	1268	1268
la37	1454	1459	1440	1397	1397
la38	1273	1248	1259	1223	1196
la39	1285	1249	1264	1233	1233
la40	1277	1355	1323	1254	1222
swv02	1655	1613	1697	1564	1475
swv05	1668	1660	1622	1600	≥1421

Table 1: Comparison of quality of solutions of the single agents vs. TECHS team

solution and due to the B&B agents it also could prove that the solutions are optimal.

One might argue that the quality differences between the solutions of the team and any member of it are not so big, so that the additional use of two workstations is a too high price to pay for the improvement. But firstly, the team is always better than its best member, so that a comparison to the competitive approach –that also would need three workstations– should be made. Secondly, in most cases the TECHS team is able to produce solutions comparable to the solution of its best member in less time. This is demonstrated in Table 2. In this table we document the times (in seconds) that the best agent working alone needed to produce its solution and the time the team needed to produce a solution that was at least as good as that of the best agent. The presented times denote the moments the solutions were produced; naturally the agents continued to work afterwards. The last column gives the speed-up factors achieved by TECHS. For 5 examples these speed-ups were less than linear, but for 6 examples they were super-linear. The best achieved factor is 55 (!) for example la37. Only for swv02 the result of Table 2 is not satisfactory, but the improvement in quality achieved by TECHS for this example (see Table 1) is very satisfactory.

It is well known that B&B search can in many cases profit from good initial solutions that help to prune the search tree. One might think that our good experimental results are only due to the NY agent providing such solutions, so that both the positive partial solutions send to this agent and the solutions send by this agent to influence the controls of the other agents are not necessary. In order to measure the effect this additional information has on the performance, we repeated the experiments without exchanging those types of information between the NY agent and the B&B agents. Despite the fact that our parameter settings drastically limited the number of pieces of information of these types, they had for many of the examples quite some positive impact on the search performance, as Table 3 shows. Only for example la27 the additional information did not pay off. For all other examples

Ex.	Best agent		TECHS team		Speed-up
	quality	time	quality	time	
la26	1218	8874	1218	4416	2
la27	1298	24444	1296	1608	15
la28	1300	2952	1298	817	3.6
la29	1241	25936	1239	2767	9.3
la36	1291	12342	1291	2104	5.8
la37	1440	26180	1426	469	55
la38	1248	14822	1248	7240	2
la39	1249	24474	1245	2516	9.7
la40	1277	2757	1273	1403	1.9
swv02	1613	6609	1608	6606	1
swv05	1622	22563	1616	7975	2.8

Table 2: Comparison of times needed to find comparable solutions

Ex.	TECHS team		restricted team	
	quality	time	quality	time
la26	1218	4416	1218	7527
la27	1271	27381	1270	27451
la28	1251	9402	1300	3140
la29	1228	3391	1241	17936
la36	1268	19552	1268	23907
la37	1397	9812	1440	26620
la38	1223	28217	1244	28271
la39	1233	10384	1233	24736
la40	1254	14752	1273	1554
swv02	1564	22778	1600	10084
swv05	1600	24521	1622	22341

Table 3: Comparison team with all type of information vs restricted team

either the achieved solution quality was better or the time to achieve this quality was (much) smaller. For examples la29, la37, and la38 we even got improvements in both categories. It should also be noted that for example la36 the time of the TECHS team is the total run-time, i.e. the time until the team can guarantee that it has found the optimum, while the restricted team only found the solution at the given time and was afterwards not able to show that it is optimal (until the allowed run-time expired). So, the exchange of information of different types is very useful and the role of a GA in a heterogeneous system is more than just producing a few initial solutions.

## 6 Related Approaches

There are many parallelization approaches to search that are based on some cooperation of search agents. Most of them, however, employ homogeneous agents, i.e. agents using the same search model. An overview for evolutionary algorithms can be found in [Ca95] and an overview of such approaches for B&B search is [GC94].

There are only a few approaches that allow to employ search agents with different search models, thus achieving cooperation of heterogeneous agents, and many of these approaches have limitations on the search models that can be used. For example, the injection island model of [Eb+97] allows the cooperation of agents employing evolutionary algorithms that differ in their fitness functions. Also the TEAMWORK approach of [De95] and [DS97] has more features of a homogeneous system than of a heterogeneous one. Search agents use set-based search models and differ either in the operators that define transitions or in the search controls, only.

The TEAMWORK approach has some similarities to the TECHS approach. The cycle of a search agent consists also of a working phase and a cooperation phase and referees play an important role in the cooperation phase. But in a TEAMWORK-based system, referees only select elements of the current state of an agent in a success-driven manner. No negative information can be exchanged. As kind of control information only a measure of success for an agent is computed. During the cooperation phase, which is synchronized for all agents, not only the referees work but also one agent has the role of the supervisor, that is a central control for the system. The supervisor collects the information selected by the referees, determines the agent with the best measure of success, and generates a new start state for all search agents by adding to the whole current state of the best agent the selected elements of the other agents. The supervisor also uses the measure of success to find agents that performed badly and replaces them with other agents.

Obviously, TEAMWORK can only be used if all search agents use a set-based search model (with the same set of possible elements). There are no receive-referees and no send-referees responsible for only a few of the other agents. Also, the control information that is exchanged is very abstract.

Another cooperation concept that is intended for mainly set-based search agents (but with possibly differing sets of possible elements) are A-Teams (see [TSM93]). The general idea of A-Teams is a cyclic data pipeline between several blackboards. A search agent uses elements from one or several blackboards as premises for its transition rules and generates new elements for a blackboard or it does not need any premises and just generates new elements for a blackboard. Some agents eliminate elements from blackboards. [BB97] presents an A-Teams-based system for JSS. Besides agents using GAs, there are also agents employing hillclimbing, simulated annealing, and tabu-search. In A-Team-based systems, agents cooperate by exchanging positive information (elements of the blackboards) only. There is no filtering involved. Typically, search agents have no own search state, all information is represented in the blackboards.

[HW93] presents an improvement of A-Teams that results in a really heterogeneous search system (like TECHS). The agents either use an or-tree-based search model or they are local optimizers. They cooperate using a blackboard on which they write so-called *hints* which are partial solutions to the

search problem. In contrast to TECHS, these partial solutions are used only as one kind of positive information and the selection of hints is not done involving referees that employ knowledge but randomly based on given probabilities. Also the integration of hints into the search of an search agent is done with a certain probability and not filtered by the demands of the agent.

## 7 Conclusion and Future Work

With TECHS, we have presented a concept that allows the cooperation of evolutionary algorithms with systems based on other search paradigms. The key differences (and improvements) of TECHS to other such approaches is the exchange of different types of information between the search systems involved and the concept of send- and receive-referees that drastically reduces the amount of communication between the systems while still selecting the important information. Our experiments with a TECHS-based system for solving job-shop scheduling problems have shown that TECHS results in synergetical improvements both in quality of the solutions found and time needed to find solutions of a certain quality. The experiments also demonstrated that all information types contributed to these synergetical improvements.

Future work will center on two directions of research. Firstly, we want to improve on our referees and also include integrating negative partial schedules into the agents based on the Genetic Algorithm. One possibility seems to be to avoid in crossovers the construction of individuals that match such negative partial schedules thus limiting the number of possible individuals. Our second direction for future research is the addition of more search agents to the system. In a first step, agents employing other branch-and-bound based algorithms or other Genetic Algorithms should be integrated. But also agents employing some other search paradigms, like tabu-search or simulated annealing, might be interesting. Finally, because we see TECHS as a general cooperation concept for search systems, other application areas should be investigated. Although our claim is already substantiated by our experiments with TECHS and automated theorem provers, other optimization problems should also be considered.

## Bibliography

- [BB97] **Bradwell, R.; Brown, K.N.:** *Application of Multi-Agent Cooperative Search To The Job-Shop Scheduling Problem*, Proc. 16th Workshop of the UK Planning and Scheduling Special Interest Group, Durham, 1997, pp. 213–218.
- [BJS94] **Brucker, P.; Jurisch, B.; Sievers, B.:** *A branch and bound algorithm for the job-shop scheduling problem*, Discrete Applied Mathematics 49, 1994, pp. 107–127.
- [Ca95] **Cantu-Paz, E.:** *A summary of research on parallel genetic algorithms*, Report 95007, University of Illinois at Urbana-Champaign, 1995.
- [CP90] **Carlier, J.; Pinson, E.:** *A practical use of Jackson's preemptive schedule for solving the job shop problem*, Ann. Oper. Res. 26, 1990, pp. 269–287.
- [DD98] **Dahn, I.; Denzinger, J.:** *Cooperating theorem provers*, in Bibel, Schmitt (eds.): *Automated Deduction – A Basis for Applications*, Vol. II, Kluwer, 1998, pp. 383–416.
- [De95] **Denzinger, J.:** *Knowledge-Based Distributed Search Using Teamwork*, Proc. ICMAS-95, San Francisco, AAAI-Press, 1995, pp. 81–88.
- [DF99] **Denzinger, J.; Fuchs, D.:** *Cooperation of Heterogeneous Provers*, Proc. IJCAI-99, Stockholm, Morgan Kaufmann, 1999, to appear.
- [DS97] **Denzinger, J.; Scholz, S.:** *Using Teamwork for the Distribution of Approximately Solving the Traveling Salesman Problem with Genetic Algorithms*, SEKI-Report SR-97-04, University of Kaiserslautern, 1997.
- [Eb+97] **Eby, D.; Averill, R.C.; Gelfand, B.; Punch, W.F.; Mathews, O.; Goodman, E.D.:** *An Injection Island GA for Flywheel Design Optimization*, Proc. EUFIT'97, 1997.
- [GC94] **Gendron, B.; Crainic, T.G.:** *Parallel Branch-and-Bound Algorithms: Survey and Synthesis*, Operations Research 42(6), 1994, pp. 1042–1066.
- [GT60] **Giffler, B.; Thompson, G.L.:** *Algorithms for solving production scheduling problems*, Operations Research 8, 1960, pp. 487–503.
- [HW93] **Hogg, T.; Williams, C.P.:** *Solving the Really Hard Problems with Cooperative Search*, Proc. AAAI-93, Washington, AAAI-Press, 1993, pp. 231–236.
- [Ka97] **Kambhampati, S.:** *Challenges in bridging plan synthesis paradigms*, Proc. IJCAI-97, Nagoya, Morgan Kaufmann, 1997, pp. 44–49.
- [NY92] **Nakano, R.; Yamada, T.:** *A genetic algorithm applicable to large-scale job-shop problems*, Proc. PPSN 2, Elsevier, 1992, pp. 281–290.
- [RS64] **Roy, B.; Sussmann, B.:** *Les problèmes d'ordonnancement avec contraintes disjonctives*, SEMA, Note D.S. No. 9, Paris, 1964.
- [SKM97] **Selman, B.; Kautz, H.; McAllester, D.:** *Ten Challenges in propositional Reasoning and Search*, Proc. IJCAI-97, Nagoya, Morgan Kaufmann, 1997, pp. 50–54.
- [TSM93] **Talukdar, S.N.; de Souza, P.S.; Murthy, S.:** *Organizations for Computer-based Agents*, Journal of Engineering Intelligent Systems, 1993.