

Parallel Metaheuristics

Teodor Gabriel Crainic

Département des sciences administratives

Université du Québec à Montréal

and

Centre de recherche sur les transports

Université de Montréal

Michel Toulouse

School of Computer Science

University of Oklahoma

and

Centre de recherche sur les transports

Université de Montréal

November 1997

Abstract

Metaheuristic parallel search methods – tabu search, simulated annealing and genetic algorithms, essentially – are reviewed, classified and examined not according to particular methodological characteristics, but following the unifying approach of the level of parallelization. It is hoped that by examining the commonalities among parallel implementations across the field of metaheuristics, insights may be gained, trends may be discovered, and research challenges may be identified. Particular attention is paid to applications of parallel metaheuristics to transportation problems.

Key words: Metaheuristics, search methods, parallel algorithms, genetic algorithms, simulated annealing, tabu search, transportation

Résumé

Nous révisons, classons et analysons des méthodes de recherche méta-heuristique parallèles – recherches avec tabous, méthodes de recuit simulé et algorithmes génétiques –, non pas selon des caractéristiques méthodologiques particulières, mais plutôt selon le critère unificateur du niveau de parallélisation. Nous pouvons ainsi faire ressortir les points communs, mettre en lumière les tendances générales, identifier les défis importants de recherche. Une attention particulière est apportée aux applications des méta-heuristiques parallèles aux problèmes de transport.

Mots clefs: Méthodes de fouille, méta-heuristiques, algorithmes parallèles, recherches avec tabous, méthodes de recuit simulé, algorithmes génétiques, transport

1 Introduction

Heuristics have been, and continue to be, an essential component of the methodological approaches used to address combinatorial optimization formulations, in general, and transportation applications, in particular. In the last ten to fifteen years, metaheuristics have profoundly changed the way we address these problems and have significantly contributed to efficiently address complex, hard problem settings (see, for example, Crainic and Laporte, 1997, or Golden *et al.*, 1998).

Parallel versions of metaheuristic methods are proposed with increasing frequency. The usual goals for parallel computing are also invoked here: reasonable computing times with more realistically formulated and sized problem instances. A second benefit is increasingly being acknowledged: in appropriate settings (e.g., multithread strategies), parallel metaheuristics may be much more robust than sequential versions relative to differences in problem types and characteristics and the corresponding parameter calibration issues.

In fact, the number of researchers and studies dedicated to parallel metaheuristics has reached the level where surveys, taxonomies and syntheses are proposed: Jog, Suh and Gucht (1991), Lin, Punch and Goodman (1994), Gordon and Whitley (1993), Gordon (1994), Cantù-Paz (1995), Hoffmeister (1991), Sargent (1988), Greening (1989a, 1990), Azencott (1992a), Lee and Lee (1992a), Verhoeven and Aarts (1995), Voss (1993), Pardalos *et al.* (1995), Laursen (1996), Crainic, Toulouse and Gendreau (1997), and Holmqvist, Migdalas and Pardalos (1997), among others. However, most of these works study parallel metaheuristics that belong to one methodology only. Even when several metaheuristic approaches are considered in the same paper, surveys are presented along exclusive methodological lines without the benefit of a global view.

The goal of this paper is to start to bridge this gap. Our approach is based on the observation that despite a large number of efficient implementations for particular problems, relatively few fundamental ideas have been used to design parallel strategies for metaheuristics. It is thus our belief that by examining the commonalities among parallel implementations across the field of metaheuristics, insights may be gained, trends may be discovered, and research challenges may be identified.

The paper is organized as follows. Section 2 presents a few statistics illustrating the increasing interest in parallel metaheuristics and presents the criteria we use to survey the literature. Sections 3, 4 and 5 are dedicated to the survey and discussion of issues related to the parallelization of metaheuristic search methods: tabu search, simulated annealing, and genetic methods, in particular. Section 6 summarizes the

conclusions derived from the survey and points to research directions and challenges. Throughout the paper, particular attention is paid to applications of parallel metaheuristics to transportation problems.

2 Parallel Metaheuristics

Compared to exact search methods, such as branch-and-bound, metaheuristic methods do not aim to systematically explore the whole solution space. Instead, they attempt to examine only parts thereof where, according to certain criteria, one believes good solutions can be found. Avoiding local optima traps and cycling, as well as making reasonably sure that the search has not overlooked promising regions, are the usual objectives of designing good metaheuristics.

Metaheuristics for optimization problems may be described summarily as a “walk through neighbourhoods”, a search trajectory through the solution domain of the problem at hand. These are iterative procedures that *move* from a given solution to another solution in its *neighbourhood*. Thus, at each iteration, one evaluates moves towards solutions in the neighbourhood of the current solution, or in a suitably selected subset. According to various criteria (objective value, feasibility, probability, etc.), a number of good (though not necessarily improving) moves are selected and implemented. Tabu search and simulated annealing methods usually implement one move at each iteration, while evolutionary methods may generate several new individuals at each generation-iteration. Moves that are evaluated at each iteration may belong to only one type (e.g., add an element to the solution) or may belong to several quite different types (e.g., evaluate both add and drop moves). Moves may marginally modify the solution or drastically inflect the search trajectory. The first case is often referred to as the *local search* phase. The diversification phase of tabu search or the application of mutation operators in an evolutionary process are typical examples of the second alternative; this last case may also be described as a change in the “active” neighbourhood.

The metaheuristics most oftenly used and parallelized are either *evolutionary approaches – genetic algorithms* (**GA**: (Holland, 1975; Goldberg, 1989; Whitley, 1994; Fogel, 1994; Michalewicz, 1992; etc.), *simulating annealing* methods (**SA**: Metropolis *et al.* (1953; Kirkpatrick, Gelatt and Vecchi, 1983; Laarhoven and Aarts, 1989; Aarts and Korst, 1989; etc.), or *tabu search* procedures (**TS**: Glover 1986, 1989, 1990, 1996; Glover and Laguna, 1993; Osman and Kelly, 1996, etc.). Other methods, such as GRASP (Feo and Resende, 1995) and *ant colony systems* (Dorigo, 1992; Coloni, Dorigo and Mannienczo, 1991, 1992) have also been proposed recently and we include them in the paper as well.

GA		SA		TS	
< 1990	\geq 1990	< 1990	\geq 1990	< 1990	\geq 1990
22	88	33	42	1	35

Table 1: Number of Publications of Parallel Metaheuristics

The continuous and rapid increase in the number of parallel metaheuristic developments and applications is illustrated in Table 1. The table synthesizes the reports, theses, and articles referenced in this paper according to the type of approach. The contributions are further separated according to the year of publication: before and after 1990. These two columns represent approximately the same time span, since most contributions before 1990 have been reported in the latter half of the 80's. The list is not exhaustive, but it yields a number of observations that will be more thoroughly addressed in this paper:

- Genetic-type methods have received the most attention. This is not surprising given the fundamental “parallel” nature of evolutionary methods.
- Many of the applications of parallel metaheuristics belong to various computer science fields: VLSI design – especially circuit placement and routing – image processing, artificial intelligence, etc. Regarding combinatorial optimization, most applications address the *travelling salesman problem (TSP)* and the *quadratic assignment problem (QAP)*.
- Parallel tabu search methods strongly emerged after 1990. Furthermore, while addressing the same general application areas previously indicated, it is parallel TS procedures that have been more often dedicated (20 out of the 36) to problems relevant to transportation science: TSP, *vehicle routing problems (VRP)*, and *network design*.
- A number of other techniques also emerged after 1990, such as parallel GRASP (Feo, Resende and Smith, 1994; Pardalos *et al.*, 1995), *ant colony systems* (Coloni, Dorigo and Mannienco, 1991, 1992; Dorigo, Manienco, and Coloni, 1996), and hybrid methods that combine elements of several methodological approaches (e.g., GA and SA).

In order to bring some order to our survey of parallel metaheuristics, independent of particular methodological characteristics, we classify them according to the level of impact

the parallelization strategy has on the algorithmic design of the metaheuristic. This classification, which has been used to study exact search methods (see, for example, Gendron and Crainic, 1994), allows us to bring together the efforts deployed according to different metaheuristic methodologies. It classifies parallel methods according to the following three strategies:

Type 1: parallelization of operations within an iteration of the solution method. This strategy, also called low-level parallelism, is rather straightforward and aims solely to speed up computations, without any attempt at a better exploration (except when the same total wall clock required by the sequential method is allowed in the parallel process) or higher quality solutions. It yields the same solution method as the sequential one, only faster.

Type 2: decomposition of the problem domain or search space. This strategy is based on the principle that computing power may be dedicated to solve a host of smaller problems, out of which an improved overall solution may be extracted or constructed. The search trajectory of the resulting parallel approach is different, however, from that of the corresponding sequential method.

Type 3: multisearch threads with various degrees of synchronization and cooperation. This strategy attempts a more thorough exploration of the solution space by initiating several search threads that simultaneously proceed through the domain.

When describing parallel metaheuristics using this taxonomy, we often refer to the performance of the methods. It is beyond the scope of this paper to address the issue of how to measure the performances of metaheuristics. The interested reader may consult Barr and Hickman (1993), Barr *et al.* (1995), and Toulouse, Crainic and Gendreau (1996). It may be useful, however, to recall that the most widely used performance measures attempt to compute some kind of acceleration. This *speed-up* is defined (relative to the definition of time) as the ratio of sequential time to parallel time to solve a particular problem on a particular machine. In notation form, let T_s be the total solution time of a sequential procedure (normally, the best one available) for a given problem and computer, and T_p the execution time of a parallel implementation with p processors. The speed-up is then $S(p) = \frac{T_s}{T_p}$. For various specifications of the serial and parallel procedures, one obtains the measures presented in Barr and Hickman (1993).

Two other performance measures are often used. *Work* (W), $W = T_p \times p$, computes the total effort required by the parallel procedure. This measure is particularly relevant when the optimum is not necessarily reached (or proven), and the quality of the solution attained by two procedures will be compared. *Scalability* measures the capability of a parallel procedure to display the same performance level when the problem size and the number of available processors grow proportionally.

3 Type 1: Low-Level Parallelization

Type 1 parallelizations correspond to the concurrent computation of some or all operations making up an iteration of the corresponding sequential method. Evaluation of individuals and moves are typical operations subject to Type 1 parallelization.

As mentioned by several authors (Crainic, Toulouse and Gendreau, 1996; Roussel-Ragot and Dreyfus, 1992; Triennekens, 1992, Gendron and Crainic, 1994), Type 1 parallelization strategies aim directly to reduce the execution time of a given solution method. In fact, when the same number of iterations are allowed to both sequential and parallel versions of the method and the same operations are performed at each iteration (e.g., the same set of candidate moves is evaluated and the same selection criterion is used), the parallel implementation follows the same exploration path through the problem domain as the serial method and yields the same solution. As a result, standard parallel performance measures apply straightforwardly.

Some implementations modify the parallel version to take advantage of the extra computing power available without altering the basic search method. For example, one may evaluate all the moves in the neighbourhood of the current solution instead of only those in a given subset. The resulting search patterns of the serial and parallel implementations are then different in most cases. Yet, because the fundamental algorithmic design is not altered, these approaches still qualify as low-level parallelism. Speed-up measures may thus be applied directly, even though the notion of *solution quality* (i.e., does the method find a better solution?) may qualify the acceleration measures.

3.1 Genetic Algorithms

In the literature, low-level parallelization of genetic algorithms is called *global* parallelization (Gordon, 1994, Cantú-Paz, 1995). Historically, this strategy was presented in what probably constitutes the first attempt to construct a parallel metaheuristic: the parallel genetic algorithms of Grefenstette (1981). In this approach, only one population is considered, and there are generally no restrictions on the selection and crossover operators used. The focus is on the parallelization of the fitness evaluation of individuals and, to a lower extent, of the crossover and mutation operators one applies to obtain a new population generation.

Parallel fitness evaluation is obtained via a synchronous master-slave approach, where the master allocates individuals to slave processors to be evaluated. There is no communication among slave processes; only the master communicates with the slaves at the beginning and end of the evaluation process. To obtain a new generation, the population is divided among processors and the usual genetic operators (selection, crossover, mutation, etc.) are applied to each partition in parallel. Note that on shared-memory computers, there are generally no explicit masters. To begin the evolution, the population is divided and each

subpopulation is assigned to a specific process. Then, a synchronization phase between generations ensures that the relevant information is available to all.

Parallel genetic algorithms of this type are described by Fogarty and Huang (1990; see also Cui and Fogarty, 1992). They used a transputer network for a pole-balancing application and reported that 1) transputer network topology is not important when computing time is more significant than communication time, and 2) communication overhead increases very fast with the number of processors.

Abramson and Abela (1992) and Abramson, Mills and Perkins (1993) implemented similar strategies for school time tabling problems on a shared-memory computer (an Encore Multimax with 16 processors) and for train timetable construction on a distributed-memory machine (a Fujitsu AP1000 with 128 processors), respectively. Almost linear speed-ups were reported for both applications using up to 16 processors, while performances deteriorated rapidly with additional processors on the distributed-memory computer.

Hauser and Männer (1994) used three different shared-memory computers – a 7-processor NERV, a 16-processor SparcServer and a 20-processor KSR1 – to implement a global parallel genetic algorithm (PGA) for a cell placement problem. Relatively good performances were reported only on the NERV multiprocessor (speed-up of 5 using 6 processors, compared to a speed-up of 2 with 8 processors on the SparcServer and 3 with 20 threads on the KSR1) due to the very low communication overhead of the machine. Recently, Chen, Nakao and Fang (1996) applied a master-slave strategy to image restoration problems using a cluster of IBM RISC6000 (from 1 to 20 machines). The best speed-up (10) was reported when 15 processors were used.

A different, asynchronous perspective is mentioned by Hoffmeister (1991) when only the fitness evaluation is to be performed in parallel. Slave processes are no longer synchronized, which may help avoid processor idle time. It does not seem, however, that this research direction has been explored any further.

It thus appears that global parallelization strategies are limited. Indeed, while fitness evaluation is a time-consuming operation and is a likely candidate for efficient parallelization, the case is less obvious concerning the genetic operators: their application is generally simpler and the extra cost of partitioning the population among processes and the resulting communications may offset the gains of parallel treatment, especially on distributed-memory architectures.

3.2 Simulated Annealing

A major issue in designing parallel simulated annealing (PSA) methods is whether the parallelization strategy affects the statistical convergence properties of the simulated annealing method. The issue has been central to all development efforts in the field and a serious

debate on whether specific parallel strategies do indeed affect the convergence properties, or whether this impact is really significant has been animating the PSA community. In the present review, we consider that PSA methods use low-level parallelism if they do not, or are assumed not to, affect the convergence of the method and do not modify the search trajectory of the sequential SA.

An iteration of the sequential simulated annealing algorithm consists of four main steps: select a move, evaluate the cost function, accept/reject the move and replace the current solution if the move is accepted. Two main approaches are used to design Type 1 parallel simulated annealing methods: *single-trial* parallelism where only one move is examined at a time, and *multiple-trial* strategies where several moves are evaluated simultaneously (Aarts and Korst, 1989; Roussel-Ragot and Dreyfus, 1992).

In single-trial methods, functional parallelism is applied to some of the computations that make up an SA iteration: one process implements the sequential SA and decomposes computing intensive tasks into smaller problems which are assigned to the other processors. Kravitz and Rutenbar (1986), who studied the cell placement problem (an important component of VLSI layout design) on multiprocessor computers, applied such a strategy and decomposed the move-evaluation step. They reported moderate speed-ups (2 on 3 processors), which they did not expect to improve with more processors. See also Bannister and Gerla (1989) for a single-trial implementation.

The single-trial strategy clearly does not alter the algorithmic design or the convergence properties of the sequential SA, which explains its initial appeal. It is, however, strongly dependent on the application and implementation. Furthermore, it tends not to speed up computations significantly. Hence, research has moved toward strategies, such as multiple-trial approaches, where a higher degree of parallelism may be attained.

In multiple-trial parallelizations, each processor executes the four steps of a simulated annealing iteration. This does not raise particular problems for the first three steps since these tasks are essentially independent for different potential moves. The replacement step is, however, a fundamentally sequential operation: only one modification operation may be executed at any given moment on the current solution. Consequently, executing this step in parallel may cause significant problems. Consider, for example, two processes that start from the same solution. Each selects a move, evaluates it and determines that it may be accepted. Each evaluation, performed concurrently with the other, assumes that the initial configuration is not changed. Then the two accepted moves are implemented, sequentially. But, as soon as one of the two moves is implemented, the evaluation of the other one is no longer valid and its implementation may lead to a configuration and cost function evaluation significantly different from what was computed by the individual process.

The concurrent execution of the replacement steps may thus yield erroneous evaluations of the cost function, resulting in a search trajectory different from the sequential one or even in violation of the statistical convergence hypotheses of the SA method. How to avoid this problem or to control the amplitude of the error introduced by the multiple-trial

parallelization strategies constitutes a central issue in the literature on parallel simulated annealing procedures.

According to our classification, multiple-trial implementations that belong to Type 1 parallelism correspond to methods where the parallel trials start from the same solution, have access to moves in the entire neighbourhood, and always result in an *error-free* cost function evaluation. The error-free evaluation is achieved because the replacement of the current solution is either restricted to a single accepted move (the others being discarded) or to moves that do not interact with each other. The latter approach is called the *serializable subset* concept, where a *serializable subset* corresponds to a set of moves that always produces the same result when applied to the current state of the system, independent of the order in which they are applied (a trivial serializable subset contains only rejected moves).

Witte, Chamberlain and Franklin (1990, 1991) studied assignment problems and proposed a move-evaluation strategy that may be compared to the probing approach in tabu search methods. Three processors were used for each move evaluation. One performs the evaluation. The others two proceed as if the evaluation result is already known: one starts from the current configuration (move rejected), the other from the configuration resulting from an eventual move acceptance. Thus, when the decision of the first processor is known, one process is killed, while for the other one some work has already been accomplished. The number of levels in the tree can grow until the pool of processors has been exhausted. While imaginative, there is much unnecessary work done (in addition to the usual communication and dispatch overhead) and performances have not been very impressive. In fact, the authors reported a speed-up of 2.6 when 8 processors were used. Nabhan and Zomaya (1995) improved somewhat the performances of this approach by modifying the information that is exchanged among processes. Only the moves actually performed to reach the new solutions are communicated, rather than the whole solutions as in the original method by Witte, Chamberlain and Franklin. Nabhan and Zomaya reported modest performance improvements for small problems (20-city TSP). The speed-up improved when the problem size was increased (almost 3.5 for 6 processors, for a 100-city TSP).

Roussel-Ragot and Dreyfus (1990, 1992; see also Roussel-Ragot, Kouicem and Dreyfus, 1990) studied the cell placement problem and experimented (for 25, 49 and 81 cells) on networks of transputers. Two network sizes were used: 3 and 6 “slaves”, plus one master that monitored the annealing schedule, chose the accepted move, and updated the memories of each processor. In the high temperature mode of the annealing schedule, all processors were synchronized after the accept/reject step and one accepted move was chosen randomly to replace the current solution. In low temperature mode, processors computed asynchronously until one processor accepted a move. Then, all processors were synchronized and updated with the new current solution. Thus, there was no error in the evaluation of the cost function. Performances were not very good. Borut and Silc (1994) also proposed an error-free PSA based on synchronization and random selection of the “new” global solution, but did not report on their implementation.

Kravitz and Rutenbar (1987; see also Rutenbar and Kravitz, 1986) studied the same

problem, and implemented an error-free multiple-trial strategy where only noninteracting moves were executed according to a simple serializable subset idea. This, however, resulted in very poor performance in high temperature mode because of the considerable amount of move cancellations.

We include these error-free multiple-trial implementations in the Type 1 parallelism category based on the assumption that these strategies, where at each iteration all processes have access to all potential moves, do not alter the search trajectory or the convergence properties of the sequential SA (multiple-trial strategies where evaluation errors occur, such as in Casotto, Romeo and Sangiovanni-Vincentelli, 1997, are classified as Type 2). The issue is not, however, altogether settled. Thus, for example, in Banerjee, Jones and Sargent (1990), the authors state that the error-free implementation of Kravitz and Rutenbar (1987) has the same convergence properties as the sequential version, while in Roussel-Ragot and Dreyfus (1990) it is stated that this same implementation “is quite different from the sequential one at high temperature since the acceptance ratio is not the same”. The debate is not, however, of great practical significance since Type 1 error-free multiple-trial strategies appear to display rather poor efficiency performances. In fact, to accept only one move implies frequent synchronizations and a lot of wasted work. Even if the number of lost moves decreases when the low temperature phase is reached, performances are significantly affected. As for the serializable subset concept, identifying a good serialization for a given subset of moves may prove to be as complex and computing-intensive as the resolution of the initial problem.

3.3 Tabu Search

Low-level parallelism has also been applied to tabu search methods (Crainic, Toulouse and Gendreau, 1997). These implementations correspond to a master thread which executes a sequential tabu procedure, but the possible moves in the neighbourhood of the current solution are evaluated in parallel by slave processors at each iteration. The slave processes may evaluate only the moves in the set they receive from the master process, or may probe beyond each move in the set. The master process receives and processes the information resulting from the slave operations, and then selects and implements the next move. The master also gathers all the information generated during the tabu exploration, updates the memories, decides when to activate different search strategies – diversification, for example – and when to stop the search.

Chakrapani and Skorin-Kapov (1992, 1993, 1995) studied quadratic assignment problems and proposed a Type 1 strategy which performs the evaluation of moves in the neighbourhood in parallel. The implementation was developed for the Connection Machine CM-2, a massively parallel SIMD machine, and was designed to take advantage of the special features of the computer. The authors reported that for problems already described in the literature, they either attained or improved the best known solutions, in a significantly smaller number of iterations. Furthermore, they were also able to determine good sub-

optimal solutions to larger problems in reasonable time. A similar strategy has also been proposed for the TSP (Chakrapani and Skorin-Kapov, 1993a).

Taillard (1991; see also, Taillard, 1993a) also addressed QAP problems and reported experimental results on a ring of 10 transputers. The set of possible moves was partitioned and each set was assigned to a different processor. Each processor then evaluated the pairwise interchange moves and identified the best one. There was no specific master processor. Instead, once a processor identified its best move, it broadcasted it to all other processors, which then performed all the normal tasks of the master: selecting and implementing the move, making the necessary adjustments and updates, partitioning the neighbourhood, etc. No implementation details were given. Load balancing through partition of the neighbourhood was acknowledged as critical, but no indication was given on how it was performed. On several problem sets proposed in the literature (essentially, the same set used by Chakrapani and Skorin-Kapov, 1993) with problem instances up to size 100, Taillard reported very good solutions, improving the best known values of many of the problems tested and obtaining suboptimal solutions (conjectured but not proven to be optimal) for problems up to size 64.

Crainic, Toulouse and Gendreau (1995) studied and compared several synchronous parallelization strategies for a tabu search procedure for the location-allocation problem with balancing requirements. Experiments were conducted on a heterogeneous network of SUN-Sparc workstations. With respect to Type 1 parallelization approaches, two variants were implemented: 1) slaves evaluated candidate moves only; 2) *probing*: slaves also performed a few local search iterations (best results obtained with 2). The second variant performed marginally better. However, both variants were outperformed by multithread (Type 3) implementations. Another Type 1, master-slave parallelization scheme is presented by Garcia, Potvin and Rousseau (1994) for the VRP with time-window constraints.

The success of Type 1 parallelizations for tabu search procedures appears more significant than for genetic approaches or simulated annealing methods. Yet, even for TS it depends heavily upon the problem characteristics. Thus, for problems where the neighbourhood is very large but the time to evaluate and perform a given move is relatively small, quadratic assignment and vehicle routing problems being classical examples, Type 1 parallelizations are very effective and near-linear speed-ups have been reported (Chakrapani and Skorin-Kapov, 1993). Performances are less interesting when the time required by one serial iteration is relatively important compared to the total solution time, resulting in executions with only a few hundred moves compared to the tens of thousands required by a typical VRP tabu search, for example. Parallel methods which attempt a more thorough exploration of the solution space than the sequential version appear to be superior (Crainic, Toulouse and Gendreau, 1995, 1995a).

4 Type 2: Parallelization by Domain Decomposition

Type 2 parallelization methods for metaheuristics are generally based on the decomposition of the decision variable vector into disjoint subsets. The heuristic procedure is then applied to each subset, the variables outside the subset being considered fixed. Such strategies are generally implemented in some sort of a master-slave framework:

- A *master* process serially devises the initial partition. During the search, it regularly modifies the partition. Modifications may be performed at intervals that are either prefixed or determined during the execution, or when restarting the method.
- *Slave* processes concurrently and independently explore their assigned partitions. The search may proceed exclusively within the partition, the other variables being considered fixed and unaffected by the moves which are performed, or they may have access to the entire solution vector.
- When slaves have access to the entire neighbourhood, the master must combine the partial solutions obtained from each subset into a complete solution to the problem.

Note that a decomposition based on the partition of the decision variable vector may leave large portions of the solution space unexplored. Therefore, in many applications, the partition is repeated to create different segments of the decision variables vector and the search is restarted.

No genetic implementations and only one GRASP procedure have been found to follow domain decomposition ideas. Therefore, this section addresses mainly simulated annealing and tabu search methods.

4.1 Genetic Algorithms

Consider an individual binary representation of length n (binary representations are used for simplicity of exposition considerations only). By fixing any set of $n - p$ bits (genes, in GA vocabulary) and stating that the remaining p bits are *free* and may take any value (this corresponds to replacing their current values with a *don't care* symbol in GA terms), one creates a *schema* (Holland, 1975; Goldberg, 1989) which represents all the possible individuals obtained by assigning specific values to the p genes. The set of all such schemata represents an implicit partition of the solution space. A parallel GA implementation based on Type 2 principles could then be built based on these schemata, according to two mechanisms. The first corresponds to sequential GA operators applied exclusively to the p free genes of each

schema. The second iteratively adapts the partition rules and selects the individuals that will yield the schemata.

We have found no parallel GA implementation that follows these mechanisms, or any other idea of individual representation partition. All the PGA we encountered in the literature that incorporate partition principles take advantage of the “natural” parallelism inherent to evolutionary techniques and work on population partitions.

We consider, however, that parallel genetic algorithms based on several subpopulations belong to Type 3 strategies. We justify this classification with the observation that even when processes are initiated with different sets of individuals, which may form a partition of some initial population, genetic operators tend to homogenize the subpopulations. In fact, because the genetic operators are not restricted to a subset of genes, one may potentially obtain any individual of the population, starting from any of the individuals of any particular subpopulation. Hence, the individual GA process is not restricted to given subpopulations; it has access to the whole population. Therefore, these methods do not work on true partitions of the variable vector or of the solution space.

4.2 GRASP

We found one Type 2 parallel implementation of the GRASP method in Feo, Resende and Smith (1994). Here, large instances of the maximum independent set problem were solved by decomposing them into many smaller problems distributed among processors and each solved by a sequential GRASP. Almost linear speed-ups were reported on an Alliant FX/80 computer with 8 processors.

4.3 Simulated Annealing

Most multiple-trial parallelization strategies for simulated annealing methods are based on the partition of the data structure. Two main approaches may be identified. The first, which corresponds to the so-called *error-free algorithms*, attempts to avoid evaluation errors and the loss of convergence properties via a strict partition of the move set and the restoration of the global state (evaluation of the objective function) following each synchronization. In the second approach, errors are admitted and the emphasis is on controlling the amount of erroneous computation.

Note that for PSA algorithms executed on shared-memory systems, it is not costly to regularly update the global state of the current solution such that errors do not accumulate during the computation. In distributed systems, however, each processor has its own copy of the data, including the “current” solution, and such global updates are costly in communication time. Thus, significant errors may accumulate locally. Trade-offs must be achieved

between the frequency of the global state update and the level of error in the PSA computation, and many studies have addressed the issue of how much error may be tolerated. (e.g., Banerjee, Jones and Sargent, 1990; Durand, 1989; Hong and McMillin, 1992; etc.).

Felten, Karlin and Otto (1985) studied the TSP and experimented with 64 cities using up to 64 processors on a hypercube computer. An initial tour configuration was randomly generated and partitioned into P subsets of adjacent cities. Each subset was then assigned to one processor. Each processor performed local swaps on adjacent cities for a number of iterations, followed by a synchronization phase where cities were rotated among processors. Parallel moves did not interact with each other due to the spatial decomposition of the decision variables, and each synchronization phase ensured the integrity of the global state. Hence, there was no error. Almost linear speed-ups were reported. A similar approach was used for the TSP by Allwright and Carpenter (1989), and by Devadas and Newton (1986) for the topological optimization of the multiple level array logic problem (on a Sequent Balance 8000 computer). Both used a similar spatial decomposition scheme and reported similar behaviours and speed-ups.

Bongiovani, Crescenzi and Guerra (1995) studied the shape detection problem and proposed an error-free PSA, strongly inspired by the particular characteristics of the problem. In this problem, given a shape represented in terms of parameters and an image, the occurrences of the shape in the image must be identified by estimating good values for the parameters. The authors formulated the problem as an optimization model aimed at minimizing the difference between the given image and the image obtained by the set of parameters, and experimented on a Encore Multimax computer with 16 processors. Their PSA used functional decomposition (single-trial, Type 1 parallelism) at high temperatures, and a multiple-trial strategy at low temperatures. An error-free computation was ensured by restricting the moves to the same sub-image.

Many of the studies proposing Type 2 parallelization approaches have been directed towards the VLSI design - cell placement problem, including Casotto, Romeo and Sangiovanni-Vincentelli (1986, 1987) who experimented (for 4, 6, 30, 101 and 122 cells) on up to 8 processors of the Sequent Balance 8000 shared-memory parallel computer. In this approach, the set of cells was partitioned into P subsets and each subset was assigned to a different processor. Moves (swapping the position of two cells) were generally restricted among cells in the same subset and were performed asynchronously. When a move involved cells on two different subsets, the processor that initiated the move synchronized with the other processor for the cell exchange. However, although the set of cells was partitioned and moves were generally restricted to the same subset, the cost function involved variables other than the cells themselves. Therefore, parallel executions of the last step of the annealing iteration interacted with each other and created errors in the cost evaluation. The authors addressed the issue of identifying partitions that reduced the errors and discussed the impact of clustering and dynamic partition modification schemes on the number of erroneous moves (between cells in different subsets). The authors also noted that, since the computation was asynchronous, the parallel search trajectory deviated from the sequential one, and that the error went to zero as the temperature decreased. This second result,

often mentioned in PSA literature, is easily explained by the fact that as temperatures decrease, fewer and fewer moves are accepted. Finally, the authors indicated that the errors introduced at high temperatures did not seem to affect the performances of the parallel method and report a speed-up close to 1. Similar observations were made (Casotto, Romeo and Sangiovanni-Vincentelli, 1987a) when massive parallelism was applied to the same cell placement problem using the Connection Machine. The authors noted, however, that the error increased with the number of moves executed in parallel.

One of the reasons for partitioning variables among processors is to prevent the same variable from being involved simultaneously in more than one move. This same goal can be achieved, however, by *locking* the variables involved in a move. In the PSA context, this mechanism allows, at any given time, only one processor, which owns the lock, to update a given variable. Any processor that attempts to execute a move involving a locked variable can either wait for it to become available or attempt a different move.

Locks have been used by Jayaraman and Rutenbar (1987) for the floorplanning problem on a distributed-memory hypercube. Processors were clustered and each cluster was assumed to implement an error-free parallel move evaluation. Locks reduced the movement of data among clusters. The access rights were modified during computation such that each cluster could have access to the whole solution space. Each cluster performed several moves before the global state was restored through synchronization. Such a “lazy” restoration strategy resulted in a significant amount of evaluation error. A maximum speed-up of 7.5 on 16 processors was reported. A similar mechanism was used in Darema, Kirkpatrick and Norton, (1987) for the cell placement problem. There was an overhead for imposing locks, however. The authors reported an overhead of 1% and 11% respectively for the two parallelization methods tested, and indicated that the overhead increased with the number of processors for both methods.

Banerjee, Jones and Sargent (1990) studied the cell placement problem, and experimented (for 32, 64, 183, 286, 469 and 800 cells) with 4 to 16 processors of a hypercube distributed-memory system. Cells were partitioned into subsets and moves interacted with each other. The issue of tolerance to error was explicitly addressed and an “integrated error control” scheme was proposed. Similar work is described in Jones and Banerjee (1987), Sargent (1988), and Rose *et al.* (1986) for the cell placement problem, as well as in Brouwer and Banerjee (1988) for the channel routing problem. Jayewardena (1990) applied the same decomposition ideas to the image reconstruction problem on a network of transputers.

Jayaraman and Darema (1988) specifically addressed the issue of error tolerance for parallel simulated annealing methods. The set of cells of a placement problem was partitioned into subsets such that two processors would not try to move the same cell. Each processor had a local view of the global state. The authors studied the impact of the frequency of synchronization and the number of processors on error tolerance. As expected, they found that the error increased as the frequency of synchronizations decreased and as the number of processors increased. The combined error created by synchronization and parallelism affected the convergence of the simulated annealing algorithm, of which parallelism emerged

as the most important factor.

A similar study can be found in Durand (1989). Vertices of the graph partitioning problem were affected to different processors of a shared-memory system. The author tested different levels of synchronization to measure the impact on the errors generated. The tests showed that PSA was tolerant to temporary error (frequent synchronizations) but at the cost of computation efficiency. On the other hand, when synchronization was relaxed, the error increased with the number of processors. An even more detailed analysis of the PSA error tolerance can be found in Hong and McMillin (1992). As for Greening and Darema (1989), they studied the partition shape and showed that it affects the cell mobility and the cost function errors. Different partition shapes can increase or reduce the frequency of synchronizations required to keep the same quality of convergence.

Greening (1990, 1990a) also studied the impact of evaluation errors on the convergence properties of the parallel SA method and analyzed the respective effects of *instantaneous* and *accumulated* errors. The author also addressed the issue of the error-free multiple-trial implementations based on spatial decomposition (e.g., Felten, Karlin and Otto, 1985; Allwright and Carpenter, 1989; Devadas and Newton, 1986). Indeed, although several authors (Banerjee, Jones and Sargent, 1990; Casotto, Romeo and Sangiovanni-Vincentelli, 1987; Darema, Kirkpatrick and Norton, 1987) considered these implementations to behave like the sequential SA, for Greening these procedures were unable to mimic the statistical properties of the sequential SA method. According to this author, the spatial decomposition “changes the pattern of the state space exploration, and thus changes the expected solution quality and execution time”. The authors of the present paper are not aware if this debate about the statistical behaviour of some multiple-trial implementations is still open in the PSA community. However, since there is no doubt that the search trajectory is indeed modified by a Type 2 parallelization, we tend to sit with Greening on this issue.

A somewhat different type of PSA implementation was reported by Banerjee and Jones (1986). The authors addressed the cell placement problem on a hypercube with the same number of processors as cells to place. Moves were evaluated in parallel by having processors on the same dimension interact. Accepted moves were then broadcasted. This method was very communication intensive: some 58% of the time processors communicated or waited for data. Furthermore, the approach appears difficult to scale. For the same problem, Rose, Snelgrove and Vranesic (1990) proposed a hybrid where the high temperature mode of the SA method was replaced with an heuristic that assigned cells to the chip. The low temperature phase was then executed in parallel.

Boissin and Lutton (1993) developed a parallel SA that can be implemented on a massively parallel computer. Experiments were performed using two combinatorial optimization problems: the QAP, and the unconstrained minimization of a 0-1 quadratic function. Interesting performances were reported on a 16K Connection Machine. Wong and Fiebrich (1987) reported another massive parallel SA implementation.

4.4 Tabu Search

Typical tabu search implementations of Type 2 parallel strategies partition the vector of decision variables directly and perform a search on each subset. This approach was part of the preliminary experimentation in the study of synchronous parallel methods for tabu search undertaken by Crainic, Toulouse and Gendreau (1995). It performed poorly, mainly because the nature of the class of problems used for testing – multicommodity location with balancing requirements – which requires a significant computation effort to evaluate and implement moves, and thus results in a limited number of moves for the entire search.

More success was been achieved by Type 2 parallel methods proposed for problems for which numerous iterations may be performed in a relatively short time, and thus restarting the method with several different partitions does not require unreasonable computational efforts. TSP and VRP formulations belong to this class of applications.

Fiechter (1994) studied the TSP. For the intensification phase of the method he proposed, each process optimized a specific slice of the tour. At the end of the intensification phase, processes synchronized to recombine the tour and modify (shift part of the tour to a predetermined neighbouring process) the partition. To diversify, each process determined among its subset of cities a candidate list of most promising moves. The processes then synchronized to exchange these lists, so that all would build the same final candidate list and apply the moves. The algorithm was implemented on a network of transputers arranged in a ring structure. The author reported near-optimal solutions to large problems (500, 3000 and 10000 vertices), and almost linear speed-ups (less so for the 10000-vertex problems).

Taillard (1993) studied parallel TS methods for vehicle routing problems. His parallelization strategies were simulated using four processors on a Silicon Graphics 4D/35 workstation for a classical set of problems ranging from 50 to 199 cities (Christofides, Mingozzi and Toth, 1979), and on a non-Euclidean 385-city problem based on actual distances and population figures from a Swiss region. The first strategy applied to Euclidean problems with uniformly distributed cities. It decomposed the domain into polar regions, to which vehicles were allocated. Each subproblem was then solved by an independent tabu search. All processors were synchronized after a certain number of iterations (according to the total number of iterations already performed), and the partition was modified: tours, undelivered cities and empty vehicles were exchanged between adjacent processors. Load balancing problems seemed to impair this approach. The second strategy was aimed at non-Euclidean problems, or problems where cities were not uniformly distributed. The main difference between the two strategies appeared in the partitioning method (the space was partitioned based on the arborescence build by the shortest paths from the depot to all cities), and in the information that is exchanged (the best solution only).

Porto and Ribeiro (1995, 1996; see also Porto, Kitajima and Ribeiro, 1996) studied the task scheduling problem for heterogeneous systems and proposed several synchronous PTS procedures where a master process determined and modified partitions, synchronized

slaves and communicated the best solutions. Several communication schemes were evaluated on an IBM 9076 Scalable POWER 1, using PVM, for varying number of processors and problem sizes. Interesting results were reported, even for strategies involving a high level of communications. Almost linear speed-ups were observed, better performances being noted for larger problem instances.

It is worth noting that currently, one of the most successful sequential metaheuristics for the VRP is a tabu search method called *adaptive memory* (Rochat and Taillard, 1995; Glover, 1996). In this approach, cities are initially separated into several subsets, and routes are built using a construction heuristic. These initial routes are then stored in a structure called an em adaptive memory. A combination procedure then builds a complete solution using the routes in the memory, and the solution is further improved using a tabu search method. The routes of “good” solutions are then deposited into the same memory, which thus adapts to reflect the current state of knowledge of the procedure. The process then starts again with a new solution built from the routes stored in the adaptive memory. The method stops when a prespecified number of calls to the adaptive memory have been performed. This approach clearly implements the principles of Type 2 decomposition using a serial procedure – see also the interesting developments in the vocabulary building strategies for tabu search proposed by Glover (1996). Adaptive memory principles are now successfully applied to other problem classes and are opening interesting research perspectives (Glover, 1996). However, interestingly enough, most parallel applications of this approach are now found in multithread strategies (Type 3).

5 Type 3: Multiple Search Strategies

Parallel methods, which consist of several concurrent searches of the solution space, are classified as Type 3 parallelization strategies. These methods may or may not use the same metaheuristic approach. They may start from the same or different initial solution and may communicate during the search or only at the end to identify the best overall solution. Communications may be performed synchronously or asynchronously, and may be event-driven or executed at predetermined or dynamically decided moments. These strategies belong to the *p-control* class according to the taxonomy proposed by Crainic, Toulouse and Gendreau (1997), and are identified as *multiple-walk* by Verhoeven and Aarts (1995).

Multiple search strategies may be further divided into two classes: *independent* and *cooperative* approaches. In the former, several searches are initiated and the best result is selected at the end from among the results of the individual processes. Independent search approaches are thus equivalent to an accelerated restarting strategy. To define a cooperative multiple search, a number of important parameters must be further decided upon (Toulouse, Crainic and Gendreau, 1996):

- the connection topology defining how processes are linked;

- the method of communication (broadcast, propagation, using a central memory, etc.)
- the processes between information exchanges are to be performed;
- the type of communication – synchronous or asynchronous;
- the time when to exchange information;
- the information to exchange.

The setting of these parameters may have a significant impact on the behaviour of the parallel procedure and the search performance. This is opening up new research avenues and we will return to this issue in the conclusion of this paper.

5.1 Genetic Algorithms

We find in the multiple search category the most widely used form of parallel genetic algorithms. Two main parallelization strategies are found: coarse and fine-grained parallelism.

For *coarse-grained* parallel genetic algorithms, parallelism is obtained by replicating the approach used in global PGA on several processors across subpopulations. Usually, the same genetic algorithm is used for all populations, although some researchers (e.g., Schlierkamp-Voosen and Mühlenbein, 1994; Herdy, 1992) have pondered the possibility of using different strategies for different populations. A *migration* operator is added to the list of genetic operators. It allows the exchange of information among subpopulations. The *emigration policy* determines how individuals are selected: best-fit, randomly, randomly among better than average individuals, etc. The *migration rate* specifies if migration involves only one individual at a time, or a pool of selected individuals. The *migration interval* determines when migration may take place, and it is usually defined in terms of a number of generations. The *immigration policy* indicates how individuals are replaced in the receiving subpopulation: worst ones dropped, random selection, random selection among the less fit individuals, etc. The information exchanges are further determined by the neighbourhood structure. In the *island* model individuals may migrate towards any other subpopulation, while in the *stepping-stone* model only direct neighbours are reachable. (In this latter approach, neighbourhoods overlap to allow for propagation of individuals). In general, the connection structure of the processors of the parallel machine on which experiments are carried out determines the connection topology defining how subpopulations are linked. Migration may be performed either synchronously or asynchronously.

Tanase (1987) proposed a stepping-stone model on 4-D hypercube topology applied to a class of Walsh polynomials (the “Tanase functions”). Migration (fixed rate, random selection from the best) occurred at regular, 5-generation intervals along one dimension of the hypercube; each subsequent migration took place along a different dimension. Two approaches were studied: same search strategy applied to all subpopulations and different

strategies. Results were reported for 2, 4, 8, 16, 32 and 64 processors; they indicated that: 1) parallel methods with different parameter values seem to perform well without knowing the best parameter settings; 2) parallel versions find results of similar quality to serial GA with near linear speed-ups.

Tanase (1989) also performed an extensive study of migration parameters. Two main conclusions emerged. First, extreme policies, such as migrating many individuals too often or few individual very infrequently, degrade performances. This result confirmed the pioneering ones of Grosso (1985). Second, even without migration, the parallel GA outperforms the serial procedure. The migration operator is further studied by Seredynski (1994) in the context of dynamic mapping and load balancing problems. Parameters such as the frequency of migrations, the number and selection strategy of the individuals for each migration, and the strategy for incorporating individuals in a new population were analyzed and tested.

Belding (1995) extended Tanase's work using the Royal Road functions (Holland, 1993) with KSR1 and KSR2 parallel computer systems. Migration destinations were chosen randomly. The author reported that, in general, the global optimum was found more often when migration was used than otherwise. Surprisingly, faster convergence was observed when the migration rate was high. The results of the study conducted by Munetomo, Takai, and Sato (1993) further emphasized the impact on the performances of distributed PGA of the frequency of exchanges among subpopulations. Similar results were reported by Kommu and Pomeranz92 (1992 who showed that with proper communication the size of each subpopulation could be substantially reduced, thus improving the computational efficiency, without decreasing the quality of the final solution.

Petty, Leuze and Grefenstette (1987) implemented a model where the best individual found at each generation in each subpopulation is broadcasted to all other subpopulations. The model was applied to functions F1 to F5 from De Jong's Test Suite (De Jong) using 1, 2, 4, 8 and 16 Intel iPSC/2 processors, for population sizes of 50, 100, 200, 400 and 800, respectively. Petty and Leuze (1989) further explored this implementation.

Cphoon *et al.* (1987) developed an island model to study the role of migration on the level of evolutionary change and evolution. The authors observed that new solutions were often found shortly after new individuals were mixed into the population. They also observed that the exchange topology was not important for the performance of the parallel GA, as long as communication lines were dense and short. They experimented with a placement problem and the parallel GA with migration outperformed the corresponding parallel GA without migration and the serial GA. Later, Cphoon, Martin and Richards (1991a,b) and Cphoon *et al.* (1991c) extended the results to various problems in VLSI and floorplan design.

Mühlenbein, Schomisch and Born (1991,b) described a stepping-stone model applied to difficult function-optimization problems. Experimentations were reported for 4 and 8 transputers. The basic strategy is quite classic: the same search strategy for all processes,

a variable migration interval and the exchange of the best individuals. The authors introduced, however, a very important addition: when a subpopulation did not improve for a certain number of generations, a local hill-climbing heuristic was applied. This approach obtained very good results. It was noted, however, that including the local optimizing procedure made it difficult to exactly determine the relative importance for the quality of the method of the heuristic and the parallel GA. The success of the approach has convinced, however, most of the PGA community and it is now widely used.

Schnecke and Vornberger (1996) proposed a PGA for the VLSI placement and routing problem using 12 transputers of the Parsytec parallel computer. Each processor executed a different GA strategy. There was no migration among the subpopulations; rather the paper emphasized the self-adaptation of the search strategies. Thus, at fixed intervals, the different GAs were ranked and the search strategies were adjusted according to the “best” one by importing some of its characteristics (mutation rate, crossover rate, etc). The paper refers to several other papers where self-adaptation strategies were developed and tested. In particular, Lis (1996) applied self-adaptation to the mutation rate. The author implemented a farming model where a master processor managed the overall population and sent the same set of best individuals to slave processors, each of which had a different mutation probability. Periodically, according to the mutation rate of the process that obtained the best results, the mutation rates of all slave processors were shifted one level up or down and populations were recreated by the master processor using the best individuals of the slave processors. Starkweather, Whitley and Mathias (1991; see also Whitley and Starkweather, 1990a,b) also suggested that an adaptive mutation rate might help achieve better results for PGA. The same authors also noted that if partial solutions could be combined to form better solutions, then a parallel GA would probably outperform a serial GA. If on the other hand, the recombination generally yielded a less-fit individual, the serial GA would outperform the PGA.

Davis, Liu and Elias (1994) applied a parallel GA to the VLSI circuit synthesis problem using 20 SPARC workstations. Although the computer architecture offered distributed memory, the implementation made use of the Linda language, which creates a virtual-shared memory programming environment for the cluster of workstations. Three parallelizations were compared: global (low-level), coarse-grained interacting and non-interacting PGA. Best solutions were obtained by the coarse-grained interacting implementation, followed by the global approach, with the non-interacting PGA obtaining solutions that were worse than those obtained with the sequential version. The three parallel versions used approximately the same computation time for a fixed number of generations.

Shonkwiler (1993) made use of independent searches. The same genetic algorithm was run on each processor (using different seeds for the random number generator), and there was no migration among the populations other than gathering the end results. The paper reports on the experimentations conducted on a cluster of SunSparc stations (2, 4 and 6 stations) on the password, the Sandia Mountain and the Inverse Fractal problems. (See also Ghannadian, Shonkwiler, and Alford, 1993 and Miller and Shonkwiler, 1992). Superlinear speed-ups were claimed and a probabilistic model was proposed to explain the performance

of the independent search method. Similar models have also been proposed for independent searches using different metaheuristics, namely by Taillard (1993) and Battiti and Tecchiolli (1992).

An interesting hierarchical coarse-grained PGA was reported by Branke, Kohlmorgen, and Schmeck (1995): each subpopulation was assigned to a cluster of processors, classical Type 1 parallelism being implemented on each cluster. Migration was permitted once the average quality of the individuals in subpopulations reached a certain level. Hence, since not all subpopulations evolved at the same speed, the ones that had attained the specified level lent processors to the other subpopulations.

Levine (1996) proposed a stepping-stone PGA for the set partitioning problem, a difficult combinatorial formulation often encountered in routing and scheduling applications, most predominantly in air crew scheduling. Each subpopulation comprised 100 individuals and migration was initiated every 1000 iterations. The best individual in a subpopulation was selected to migrate and it replaced an individual selected in the receiving population, by a probabilistic tournament. Each subpopulation exchanged individuals with its four neighbours (subpopulations were arranged in a two-dimensional toroidal mesh), alternating among them each migration. Experiments were conducted using from 1 to 128 processors (by steps of powers of 2) on an IMP SP with 128 nodes, each of which was an IBM RS/6000 Model 370. Several medium problems (few rows, less than 3000 columns) and a few medium to large problems (8000 to 45000 columns and 400 to 823 rows) were used for testing. Optimal or near optimal solutions were obtained for the medium problems. No integer solution was found on problem instances with many constraints. It was observed that increasing the number of subpopulations was beneficial to the quality of the solution.

Fine-grained parallelizations are asynchronous methods that divide the population into a large number of subsets. Ideally, subsets are of cardinality one, each individual being assigned to a processor. Each subset is then connected to several others in its neighbourhood (the *deme*) and the genetic operators are applied through asynchronous exchanges between individuals in the same deme only. The deme may be of fixed topology (consisting of individuals residing in particular processors) obtained by a random walk or by applying a given Hamming distance, etc. An individual is then selected from the deme and a crossover operation is performed with the original individual. Selection may be based on various criteria: local fitness distribution, tournament, local ranking, etc. Neighbourhoods generally overlap to allow propagation of individuals or individual characteristics and mutations. Such methods are also identified as *massively parallel*, because ideally a processor is assigned to each individual, and as *cellular models*, because a parallel GA with a fixed topology deme and a relative fitness policy may be shown to be a finite cellular automata with probabilistic rewrite rules and an alphabet equal to the set of strings in the search space (Whitley, 1993).

Manderick and Spiessens (1989) experimented with a fine-grained GA on functions F1 to F5, using a Symbolics Lisp Machine. The individuals of the population were placed on a planar grid and a small, fixed-size neighbourhood was assigned to each individual. This neighbourhood usually consisted of an individual's neighbours on the grid. The application

of selection and crossover operators was restricted to the neighbourhoods, contrary to a sequential implementation. In the implementation of the algorithm on the DAP parallel computer (1024 processors) described in Spiessens and Manderick (1990, 1991), all the individuals were updated in parallel, if the population was not larger than the number of processors.

Mühlenbein, Gorges-Schleuter and Krämer (1987, 1988) applied a similar approach to the TSP using an Encore computer, which is a shared-memory system. The size of the neighbourhood was fixed to 4 neighbours, a local selection strategy was used, a hill-climbing heuristic was applied to individuals, and GA operators were applied to the resulting local optimum. Mühlenbein (1989) applied the same basic strategy to solve the QAP, but divided the individuals into two equal subsets placed on two rings. Each individual thus had two neighbours on each ring. A 64-processor transputer network was used. Later, Mühlenbein (1991c) applied the same fine-grained PGA approach to the graph partitioning problem. In the latter paper, the author suggested that different hill-climbing strategies could be applied concurrently, but did not implement this strategy. See also Laszewski and Mühlenbein (1991). Interesting overviews of Mühlenbein's views on parallel genetic algorithms, their design, the role of hill-climbing heuristics and their applications may be found in Mühlenbein (1991, 1992 and 1992a).

Gorges-Schleuter (1989, 1991, 1991a) studied fine-grained PGA and applied the strategy to the TSP using 64 processors of a transputer network. According to the author, similar to complex physical systems, the global behaviour of genetic algorithms might be better understood when viewed as an emergent phenomena resulting from the self-organization of simple and locally interacting rules. Similar to Mühlenbein (1989), individuals were divided into two equal subsets and placed on two rings. Demes were of size 8, and composed of neighbouring nodes on the two rings – the demes were thus overlapping. The selection was simple: one of the offspring replaced the parent. The mate for the crossover operation was chosen either by proportional or linear ranking selection (Goldberg and Deb, 1991). Several survival strategies (offspring replacing their parents) were tested and the strategy that accepted only offspring fitter than the local parent yielded the best solution, the best median solution and the lowest deviation.

Gorges-Schleuter also compared the similarity, with respect to the genotype, of demes that were at a physically different distance from one another. It was reported that the difference in the average Hamming distance between demes increased as the demes were more physically distant. This proves the existence of “niches” in fine-grained PGA. The author then described the impact of linear versus planar population structures on the gene pool. He reported that the fast propagation of the planar structure allowed advantageous genes to quickly overtake the whole population, but the very best solutions were missed. In Gorges-Schleuter (1992), the author continued his study of the fine-grained PGA, and examined the impact of different local mating strategies on the formation of niches. A one-gene, two-allele model of individuals was used, and there were no selection or mutation operators. Results showed that mate selection with one fixed parent was more capable of producing stable local islands.

Maruyama, Hirose and Konagaya (1993) introduced an asynchronous fine-grained PGA applied to the graph partitioning problem using a cluster of workstations and a Sequent Symmetry computer. The authors attempted to adapt fine-grained PGA to coarse-grained parallel computer architectures. Each processor had an active individual and a buffer of several suspended individuals. Active individuals were sent to all other processors. Processors then randomly selected one individual among those received from the other processors, the new individual replacing one of the suspended individuals according to the fitness function. Crossover consisted of replacing part of the active individual by a part of one of the suspended individuals. In an unusual approach, only one offspring was produced, the other parts of the active and suspended individuals being rejected. Mutation was then applied and the modified active individual was compared to the suspended individuals. If it could not survive, it was then replaced by one of the suspended individuals according to the fitness function. Tests were performed with 15 processors for the Sequent Symmetry and 6 processors for the cluster of workstations. The authors reported near linear speed-ups for the same quality of solution on both types of coarse-grained architectures. A similar approach was proposed in Maruyama, Konagaya and Konishi (1992), with the difference that the buffer for each processor contained only individuals received from the other processors.

A similar “logical” fine-grained algorithm for coarse-grained computers was proposed by Tamaki and Nishikawa (1992), for the job shop scheduling problem. It is a logical parallel method because it was first implemented on a sequential computer, then implemented on 6 and 12 transputers to speed up the computation. The demes were composed of individuals at a Hamming distance of 1. Selection was based on the local fitness distribution, crossover chose a mate randomly in the neighbourhood, and mutation was performed using one bit selected randomly. Implementing fine-grained PGA on coarse-grained computers was also one of the objectives of Voigt, Santibáñez-Koref and Born (1992; see also Boigt and Born, 1990). The algorithm was applied to the F6 function using transputers. There was an initial level of selection and reproduction operators in the local environment; these local environments being linked according to the interconnection structure of the coarse-grained computer. Then, there were selection and reproduction policies for the interactions among local environments. The authors experimented with 1 to 7 processors and different fixed topologies to define the demes: ring, torus, lattice, hypercube, etc. Hill-climbing was applied when an individual did not improve for a given number of generations. Selection used a ranking scheme, and the offspring replaced the worst neighbour if its fitness was better. The best results were obtained for ring local environments with 3 individual demes. Additional fine-grained, cellular, PGA were proposed by Sannier and Goodman (1987), Talbi and Bessière (1991, 1991a), Muntean and Talbi (1991), etc.

Collins and Jefferson (1991) developed a fine-grained PGA for the multilevel graph partitioning problem using the massively parallel Connection Machine. The main goal of the paper was to characterize the difference between panmictic (i.e., at the level of the entire population) and local selection/crossover schemes. It is known that panmictic selection/crossover converge on a single peak of multimodal functions, even when several solutions of equal quality exist. Tests were performed on a function with two optimal solutions, and the panmictic approach never found both solutions. The authors noted that

modifications to the panmictic selection and crossover operators that changed this behaviour made use of global information which was not suited for parallel implementation. On the other hand, local selection and crossover displayed an obvious parallelism. In these implementations the demes were made of a random walk on a 1- or 2-dimensional grid, the deme size being a function of the length of the random walk. Selection was local from the random walk and used local fitness distribution and local linear ranking. The experimentation used 16000 processors, for a population size of 2^{13} to 2^{19} . The method using local selection and crossover consistently found both optimal solutions, was resistant to premature convergence because each deme could have explored different peaks, found optimal solutions faster, and was more robust. Gordon (1994) further compared the natural parallelism available in global, coarse-grained and fine-grained PGA. See Maniezzo (1993) for another example of fine-grained parallel implementation on the CM@ Connection Machine.

Schwehm (1992) implemented a fine-grained PGA on a massively parallel computer, the MasPar MP-1 using 1024 of the 16384 processors. The author investigated which network topology was best-suited to fine-grained PGA. The diameter of the network determined how long it took for good solutions to propagate over the entire environment. Long diameters isolated phenotypes with little chance of combining good alleles. Short diameters prevented genotypes to evolve since good solutions soon dominated, which lead to premature convergence. The author tested the following 5 topologies: ring of 1024 elements; torus of 32×32 elements; 3-cube of $16 \times 8 \times 8$ elements; 5-cube of 4^5 elements, 10-cube of 2^{10} elements. The torus showed the best results: higher and lower dimensionality resulting in slower convergence. A similar study on the interconnection topologies of fine-grained PGA may be found in the papers of Baluja (1992, 1993). According to Baluja, the overlapping populations of the cellular PGA corrects the drawback of coarse-grained PGA where convergence of a subpopulation to an equilibrium state may prevent the incorporation of new material because of its incompatibility with existing information. However, subpopulations of cellular PGA can also be dominated by the genotype of strong individuals. Three different topologies were studied regarding their capability to prevent this problem. Numerical results suggested that 2D array topologies obtain the better results. The author also suggested that a more rigorous model of the interactions in PGA is needed since we still do not fully understand the complexity added by these interactions to the behaviour of this particular metaheuristic.

Few authors compare the respective performances of coarse and fine-grained parallel genetic algorithms, and conclusions are generally not clear-cut (Cantù-Paz, 1995). For example, Baluja (1993) compared one coarse-grained and three fine-grained algorithms and found the latter to perform better, while the tests conducted by Gordon and Whitley (1993) indicated opposite conclusions. Baluja argues that the difficulty in defining a performance measure equally adaptable to the two parallelization types plays a major role in our inability to conclude on their relative performance.

5.2 Simulated Annealing

Numerous efforts to develop Type 3 parallel simulated annealing strategies, either independent or cooperative, are reported in the literature. A very interesting development in PSA consists in the systematic inclusion of principles and operators from genetic algorithms in simulated annealing multithread procedures. These hybrids tend to perform very well.

Aarts *et al.* (1986) proposed the first Type 3 parallelization scheme for simulated annealing, called *division strategy* by Aarts and Korst (1989). Rather than having several processors execute moves from the same current solution or from a same subset of decision variables, processors worked independently on different short Markov chains. Let L be the length of the Markov chain (number of iterations) executed by an SA program before reaching equilibrium at temperature t . The division strategy consisted of executing L/P SA iterations on P processors at temperature t . At the L/P -th iteration, the processors synchronized and one solution was chosen to be the initial configuration for the next temperature. Hence, this was a synchronous cooperative scheme with global exchange of information at each interaction. Unfortunately, the length of the chain could not be reduced arbitrarily without significantly affecting the convergence properties. This was particularly true at low temperatures where a large number of steps was required to preserve equilibrium. Consequently, at low temperatures, the processors were clustered and in each cluster a parallel move strategy (Type 2) was applied. Monien, Ramme and Salmen (1995) presented another example of this last strategy.

This strategy has been applied by Diekmann, Lüling and Simon (1993) to the TSP, the graph partitioning problem and the link assignment problem, using a network of transputers. The selection of the initial configurations was based on the Boltzmann distribution. A speed-up of 85 over 121 processors was reported. According to the authors, the convergence behaviour and the quality of solution were similar to the sequential algorithm and independent of the number of processors. In their study of this type of parallelization strategy, Aarts *et al.* (1986) had predicted an upper bound of 30 useful processors after which, the speed-up and the quality of solution were supposed to decrease. Apparently, the results of Diekmann, Lüling and Simon prove that the prediction was too pessimistic.

Interactions among processors are not limited to being executed only at the end of each Markov chain. In fact, there is a wide variety of possibilities here. The simplest approach consists of only one interaction at the end. This corresponds to the independent search model studied by Azencott (1992). Lee and Lee (1992a,b) also examined this approach, as well as variants where the SA threads interact periodically (the periods may be fixed or dynamic) in synchronous or asynchronous modes. They used 2, 4, 8 and 16 IntelPSC/2 processors, and a 32-processor BBN Butterfly GP1000 computer. For the graph partitioning problem, dynamic interval exchange strategies generally performed best. Also, asynchronous strategies where communications proceeded through a central memory (blackboard architecture) obtained solutions of equal or better quality compared to synchronous parallel schemes.

According to Laursen (1994), how to schedule these interactions is still an open problem. The author proposed to use the selection and migration operators of PGA to help address this issue. On each processor there were k simulated annealing procedures and all performed concurrently for l iterations. After the l iterations, processors were paired and each migrated (copied) a number of states to its pair. After migration, each processor had $2k$ states, and this number was reduced to k by selection. These new k states were the initial configurations from which the k concurrent simulated annealing processes on each processor restarted the search. Experiments have been conducted for the QAP, the graph partitioning problem and the weighted vertex cover problem using a network of 16 transputers. The author tested three migration strategies: no migration, global and local (stepping-stone). Global migration consisted of bringing the best states to a given processor, which then chose the best among the best and broadcasted them to all processors. This strategy suffered a 10% to 25% overhead in communication cost and produced very bad solutions. The no communication (migration) approach was the fastest strategy but produced lower quality solutions compared to the local migration strategy, which incurred a 2% overhead.

Another PSA division strategy with coordination inspired from the GA operators is presented in Mahfoud and Goldberg (1995). In their paper, n Markov chains were evaluated concurrently. The SA-GA hybrid algorithm proceeded as follows: after $n/2$ iterations, two parents were selected from the population of the n current solutions. Two children were generated using a GA crossover operator, followed by a mutation operator. Probabilistic trial competitions (i.e., the winning probability depends on the current temperature and the relative difference between the fitness evaluation of the individuals that are competing with each other; several such competitions may be defined according to the exact winning probability formula, as well as the exact competition rules: one on one, the parents combined versus one of the offspring, etc.) were held between the children and the parents and the replacement step was performed according to the outcome of the competition. The temperature was lowered when the population reached equilibrium. The parallelization of this algorithm is straightforward. It divided the population of n Markov chains into P subpopulations of n/P Markov chains. Crossover, mutation and probability trials were applied to individuals of each local subpopulation. Two prototypes were implemented on the CM-5 Connection Machine, one synchronous and one asynchronous. In synchronous mode, the migration operator periodically sent the subpopulations (distributed memories) to a master processor that randomly redistributed the individuals to different processors. In asynchronous mode, processors were periodically paired and $(n/P)/2$ individuals were exchanged.

A different approach to building genetic-annealing parallel methods was proposed by Ram, Sreenivas and Subramaniam (1996). The authors described two PSA methods and applied them to the job shop and travelling salesman problems, using a network of 18 SUN workstations. Both methods were composed of two phases. The second phase of each method consisted of an independent search process, which was the same for both methods. In the first phase of the first approach, the search threads started from different initial solutions, performed a number of iterations and exchanged their respective best solutions; all threads then restarted from the overall best solution. In the first phase of the second

method, a genetic method was used to obtain a population of good solutions that was then used to initiate the independent search phase of the method. The results tended to show that given a fixed time for the first phase, the quality of the solution determined by the GA decreased as the problem size increased. When more time was allocated to the GA phase, the overhead of the parallel method became too high. Moreover, when the number of processors was increased, the number of good solutions in the population was less than the number of processors, hence some were initialized with “bad” solutions. Other GA-SA hybrids have been proposed by Sirag and Weisser (1987), Brown, Huntley and Spillane (1989), Lin, Kao and Hsu (1991), Boseniuk and Ebeling (1987), Goldberg (1990), 1988 and 1991) and Varanelli and Cohoon (1995)).

An innovative approach to developing PSA methods has been proposed by Fleischer (1996; see also Fleischer and Jacobson, 1996). The author made use of the control theory to devise a negative feedback mechanism to control the modifications to the temperature parameter and thus to the transition probabilities. The mechanism is based on probabilistic information regarding the quality of the current solutions as evaluated by two or more SA procedures that proceed concurrently and synchronously. These ideas were tested on the independent set and vertex cover problems with encouraging results. It is noteworthy that this approach does not adversely impact the convergence properties of the simulating annealing method.

5.3 GRASP

A Type 3 parallelization for GRASP is proposed by Pardalos, Pitsoulis and Resende (1995) for the QAP. It is an independent thread method implemented on a Kendal Square KRS-1 with 128 processors, 64 of which were used in the experiment. An excellent average speed-up (62) was reported.

5.4 Tabu Search

Type 3 parallelizations for tabu search methods follow the same basic pattern: p threads search through the same solution space, starting from possibly different initial solutions and using possibly different tabu search strategies. In their taxonomy, Crainic, Toulouse and Gendreau (1996) identify these various possibilities as Single (initial) Point Different Strategies (SPDS), Multiple Point Single Strategy (MPSS), and Multiple Point Different Strategies (MPDS). The search threads may proceed in a totally independent fashion, the best solution being identified at the end. These strategies are identified as *independent multithread* methods. When information is exchanged among tabu threads, the so-called *cooperative multithread* methods, synchronous communications have been mainly implemented. Asynchronous procedures are, however, increasingly being developed. Consequently, one observes an increased awareness of the issues related to the definition and modelling of

cooperation.

Battiti and Tecchiolli (1992) used the QAP to present a tabu search with a hashing procedure used to cause the search to react to the detection of cycles by suitably modifying the length of the tabu lists. The authors then analyzed a parallelization scheme where several independent search processes started the exploration of the domain from different, randomly generated, initial configurations. The authors then proceeded to derive probability formulas for the success of the global search that tended to show that the independent search parallelization scheme was efficient – the probability of success increased, while the average success time decreased with the number of processors – provided the tabu procedure did not cycle.

Taillard also studied parallelization strategies that perform many independent searches, each starting with different initial solutions. The main study is found in his paper on parallel tabu methods for job shop scheduling problems (Taillard, 1994). For this type of problem, Taillard showed that a tabu search approach, which includes a diversification phase, is very competitive. It is simpler to implement and generally more efficient than either the simulated annealing or the shifting bottleneck procedures (the two best heuristics proposed at the time). It helped establish new best known solutions for every problem in two sets of benchmark problems, while optimally solving random problems with m machines $\ll n$ jobs (e.g., $m = 5$, $n = 2000$) in polynomial mean time. Several parallelization ideas focussing on speeding up computations related to the neighbourhood evaluation (Type 1 parallelism) did not yield good results, either because the available computing platforms (a ring of transputers and a 2-processor Cray computer) were not suitable for the implementations, or because the communication times were much higher than the computation ones. Taillard then proceeded to examine the theoretical bases of the independent multithread parallelization approach for “random” iterative algorithms (tabu search, simulated annealing, etc.). His results showed that the conditions needed for the parallel approach to be “better” than the sequential one are rather strong, where “better” was defined as the probability of the parallel algorithm achieving success with respect to some condition (in terms of optimality or near-optimality) by time t being higher than the corresponding probability of the sequential algorithm by time pt . However, the author also mentioned that, in many cases, the empirical probability function of iterative algorithms was not very far from an exponential one and that the independent multithread parallelization approach is very efficient. The results for the job shop problem seem to justify this claim.

Rego and Roucairol (1996) proposed a tabu search approach for the VRP based on ejection chains, and implemented an independent multithread parallel version where each thread used a different set of parameter settings but started from the same solution. The method was implemented in a master-slave setting, where each slave executed a complete sequential tabu search. The master gathered the solutions found by the threads, selected the overall best, and reinitialized the threads for a new search. Low-level (Type 1) parallelism was used to accelerate the move evaluations of the individual searches, as well as a post-optimization phase. Experiments were conducted on a network of four SUNSparc workstations and showed the method to be competitive on a set of standard VRP problems

(Christofides, Mingozzi, and Toth, 1979).

Malek *et al.* (1989) implemented and compared serial and parallel simulated annealing and tabu search algorithms for the travelling salesman problem. The authors reported that the parallel tabu search implementation outperformed the serial one, and consistently produced comparable or better results than sequential or parallel simulated annealing. The parallel experiments were performed on a 10-processor Sequent Balance 8000 computer, for the 25, 33, 42, 57 and 100-city problems with proven optimum solution, and for the 50 and 75-city problems where only best-know solutions are available. The implementation proceeded with one main process and four child processes. Each child process ran a serial tabu search algorithm with different tabu conditions and parameters. The child processes were stopped after a specified time interval, the solutions were compared, and bad areas of solution space were eliminated. The child processes were then restarted with a good solution and an empty tabu list. Note that, in order to strictly implement this strategy, the diversification long-term memory function was disabled.

De Falco *et al.* (1994) and De Falco, Del Balio and Tarantino (1995) studied the QAP and the mapping problem, respectively, and experimented on transputer networks (with 16, 32 and 64 processors), a MAsPAR MPP-1 SIMD computer, and a Convex Meta Series MIMD machine. They implemented a multithread strategy, where the best solutions from individual processes were exchanged at each iteration among their neighbours. At each iteration, each process performed a local search from its best solution. Then, processes synchronized and the neighbours exchanged their respective best solutions; local best solutions were replaced with imported ones only if the latter solutions were better. The authors indicated that they obtained better solutions when cooperation was included compared to an independent thread strategy. Superlinear speed-ups were reported.

A continuously increasing number of asynchronous cooperative multithread search methods is being proposed. All such developments we have identified, use a *central memory* for inter-thread communications. Each individual search thread starts from a different initial solution and generally follows a different search strategy. Exchanges are asynchronously performed and proceed through the central memory.

As far as we can tell, Crainic, Toulouse and Gendreau (1997) proposed the first such strategy for tabu search as part of their taxonomy of parallel TS methods. The authors also presented a thorough comparison of various parallelization strategies based on this taxonomy (Crainic, Toulouse and Gendreau, 1995 and 1995a). The authors implemented several Type 1 and 2 strategies, one independent multithread approach, and a number of synchronous and asynchronous cooperative multithread methods. The multicommodity location problem with balancing requirements was used for the experimentation conducted on a 16-transputer network. The authors report that the parallel versions achieved better quality solutions than the sequential TS and that, in general, asynchronous methods outperformed synchronous ones. The independent threads and the asynchronous cooperative approaches offered the best performances.

An interesting approach to the development of Type 3 cooperative multithread parallel tabu search methods is based on *adaptive memory* ideas (see Section 4). It has been particularly used for real-time routing and vehicle dispatching problems (Gendreau *et al.*, 1996), as well as for VRP with time window restrictions (Taillard *et al.*, 1995; Badeau *et al.*, 1995). A general implementation framework would have each thread construct an initial solution and improve it through a tabu search or any other procedure. Each thread deposits the routes of its improved solution in the adaptive memory. It then constructs a new initial solution out of the routes in the adaptive memory, improves it, communicates its routes to the adaptive memory, and so on. A “central” process manages the adaptive memory and ensures communication among the independent threads. It also determines when the procedure stops based on the number of calls to the adaptive memory, the number of successive calls without improving the best solution, or a time limit. In an interesting development, Gendreau *et al.* (1996) also exploited parallelism within each search thread: the set of routes was decomposed along the lines proposed in Taillard’s work (1993). Very interesting performances have been observed on a network of workstations, especially when the number of processors is increased.

Crainic and Gendreau (1997) proposed a cooperative multithread PTS for the fixed cost, capacitated, multicommodity network design problem. The individual tabu search threads differ in their initial solution and parameter settings. Communications are performed asynchronously through a central memory device (blackboard implementation). The authors compared five strategies to retrieve a solution from the pool when requested by an individual TS thread. The strategy that always returns the overall best solution displayed the best performances when few (4) processors were used. When the number of processors increased, a probabilistic procedure, based on the rank of the solution in the pool, appeared to be best. The parallel procedure improved the quality of the solution and required less computing time compared to the sequential version, particularly for large problems with many commodities (results for problems with up to 700 design arcs and 400 commodities are reported). The experimental results have also emphasized the need for the individual threads to proceed unhindered for some time (e.g., the first diversification move) before initiating exchanges of solutions. This ensures that local search histories can be established and good solutions can be found to establish the central memory as an *elite candidate* set. On the other hand, early and frequent communications yielded a totally random search that proved to be quite ineffective. The authors finally reported that the cooperative multithread procedure also outperformed an independent search strategy that uses the same search parameters and starts from the same initial points. Other implementations of asynchronous cooperative multithread PTS methods were presented by Andreatta and Ribeiro (1994) and Aiex *et al.* (1996; see also Martins, Ribeiro, and Rodriguez, 1996) for the problem of partitioning integrated circuits for logical testing.

Crainic and Gendreau (1997a) report the development of a hybrid combining their cooperative multithread PTS method with a genetic engine. The GA initiates its population with the first elements of the central memory of the PTS. Asynchronous migration (rate = 1) subsequently transfers the best solution of the GA to the PTS, and other solutions of the pool towards the GA. The hybrid appears to perform well, especially on larger problems

where the best solutions known are improved. It is noteworthy that the GA alone was not performing well and that it was the parallel TS procedure that identified the best results once the genetic method contributed to the quality of the central memory.

Multithread parallelization thus appears to offer very interesting perspectives for metaheuristics. However, several issues remain to be addressed.

Synchronous implementations, where information is exchanged at regular intervals have been reported for the three classes of metaheuristics examined in this paper. In general, these implementations outperform the serial methods in solution quality. For tabu search (Crainic, Toulouse and Gendreau, 1995) and simulated annealing (Graffigne, 1992) synchronous cooperative methods appear to be outperformed, however, by independent search procedures. Yet, the study by Lee and Lee (1992a) reversed this trend. Their results show the independent thread approach to be outperformed by two strategies of synchronous cooperating parallel threads. This points to interesting research issues which should be further investigated since Lee and Lee used a dynamically adjusted synchronization interval that modified the traditional synchronous parallelism paradigm. This is also the case for genetic algorithms: Cohoon *et al.* (1987) and Cohoon, Martin and Richards (1991a, 1991b) report that parallel search with migration operators applied at regular intervals outperform the same method without migration. Asynchronous cooperative multiple search strategies appear to have been less studied. Our group has undertaken the study of distributed multithread tabu search methods (Crainic, Toulouse and Gendreau, 1995, 1996). The results, which are already available, seem to indicate that these approaches offer better results than synchronous and independent searches, but more theoretical and empirical work is still required.

An important issue that is beginning to be acknowledged in the parallel metaheuristic community, concerns the definition of cooperation schemes and their impact on the search behaviour and performance. A number of basic communication issues in designing multithread parallel metaheuristics are discussed by Toulouse, Crainic and Gendreau (1996). A more thorough analysis tends to show (Toulouse, Crainic and Sansó, 1997a, 1997b) that cooperative parallel metaheuristics form dynamic systems and that the evolution of these systems may be more strongly determined by the cooperation scheme than by the optimization process. This points to the urgent need to better understand cooperative procedures and the impact of communication parameters and design on their behaviour.

6 Conclusions and Perspectives

Metaheuristic parallel search methods, particularly tabu search, simulated annealing and genetic algorithms, were reviewed, classified and examined according not to particular methodological characteristics, but following the unifying approach of the level of parallelization. To our knowledge, it is the first time that such a review has been attempted and, although

not exclusive, it examined and identified the commonalities among parallel implementations across the field of metaheuristics.

It thus appears clearly that beyond the often very clever implementations that take advantage of the particular characteristics of the problem and computer at hand, a relatively limited number of parallelization strategies have been proposed.

All types of parallelization approaches have their merits and will certainly continue to be applied when appropriate. One may identify, however, a rather clear trend in the development of such methodologies: from low-level parallelization of computing-intensive tasks, to domain decomposition, to multithread methods where several searches explore the problem space concurrently.

Multithread approaches are increasingly proving their worth. These methods allow to increase the variety of the search threads. Not only by a more efficient implementation of restarting strategies, but more particularly by the possibility of having different types of searches – same method with different parameter settings or even different metaheuristic and exact searches – proceeding concurrently. Thus, a more thorough exploration of the solution space of a given problem instance may be obtained. As an additional benefit, multithread methods appear more robust than their sequential counterparts relative to the differences in problem types and characteristics. Such approaches also offer a relatively easy way of harnessing the power of the networks of workstations present in most firms and organizations. It is our belief that the field of multithread parallel metaheuristics, and particularly, cooperative methods, will continue to grow and offer exciting research possibilities.

Of prime importance are the issues related to the definition, understanding and control of cooperation. We have barely begun to explore this field and we realize that cooperative parallelism sometimes obeys laws that we do not fully comprehend, and that the parallel search we build may evolve by following its own dynamics and not according to our optimization goals. We need a better understanding of these dynamics and of the related global search behaviour of cooperative search, of the impact of the various design parameters on this behaviour, and of the means available to control the search trajectory. A few research efforts have been initiated, but the field is still largely unexplored.

An interesting research issue, which has been addressed by the GA and SA communities but has been largely neglected regarding tabu search, concerns the search properties of the parallel method, particularly cooperative ones, with respect to the basic hypotheses of the methodology. Thus, there has been little research dedicated to the study of parallel TS viewed as one “meta” tabu search method: How do we recreate global memories out of the individual recollections of the individual threads? How do we direct the parallel process into intensification and diversification phases? How do we make the most of the information being exchanged and thus turn the communication search overhead into an improving factor? etc. (With respect to this last question, it is worth remembering that the exchanged solutions form a type of elite “population”.) By addressing these issues, one

may not only improve the overall performance of the parallel TS procedure, but also achieve a better understanding of the cooperation process.

Hybrid methods appear to offer very interesting perspectives. They may dramatically improve the performance of some metaheuristics, even in their sequential version. We witness increasingly ideas of one methodology (e.g., GA) being used in another area (e.g., SA). We believe this holds a promising future. Similarly, the cooperation among search threads of different types of metaheuristics or exact searches (e.g., branch-and-bound and tabu search in parallel) opens up attractive perspectives.

Finally, here are a few other preoccupations and promising research questions related to parallel metaheuristics: respect to cooperative search procedures. One does not always compare the parallel procedure to the appropriate sequential one, which results in sometimes inaccurate speed-up claims. (ii) More comparisons among different types of metaheuristics are needed. (iii) More attention should be paid when designing parallel procedures for the load balancing issue. This is especially relevant for Type 1 and 2 implementation where “slave” processors may easily become idle and work for very different spans. (iv) Software libraries of parallel procedures and test problems should be built. These already exist or are being undertaken for some problem classes and mainly for sequential methods. Their existence for parallel procedures would greatly enhance our ability to construct new procedures and to evaluate and compare existing ones more thoroughly. (v) Ant colony systems are “naturally” parallel procedures but true parallelizations have yet to be attempted.

Thus parallel computation appears to be a most-promising research and development area for metaheuristics. It already efficiently addresses large instances of complex problems and holds great promises for the future. Many questions and challenges still exist but, as this review shows, a more unified view of parallel metaheuristics may significantly help to address them.

7 Acknowledgments

We would like to thank the anonymous referees whose comments have helped us write a better paper. Funding for this project has been provided by the Natural Sciences and Engineering Council of Canada, through its Research and Strategic Grant programs, and by the Fonds F.C.A.R. of the Province of Québec.

References

- [1] E. Aarts and J. Korst. *Simulated Annealing and Boltzmann Machines*. John Wiley & Sons, New York, NY, 1989.
- [2] E.H.L. Aarts, F.M.J. de Bont, J.H.A. Habers, and P.J.M. van Laarhoven. Parallel Implementations of Statistical Cooling Algorithms. *Integration, The VLSI Journal*, 3:209–238, 1986.
- [3] D. Abramson and J. Abela. A Parallel Genetic Algorithm for Solving the School Timetabling Problem. In G. Gupta and C. Keen, editors, *15th Australian Computer Science Conference*, pages 1–11. Department of Computer Science, University of Tasmania, 1992.
- [4] D. Abramson, G. Mills, and S. Perkins. Parallelization of a Genetic Algorithm for the Computation of Efficient Train Schedules. In D. Arnold, R. Christie, J. Day, and P. Roe, editors, *Proceedings of the 1993 Parallel Computing and Transputers Conference*, pages 139–149. IOS Press, 1993.
- [5] P. Adamidis and V. Petridis. Co-operating Populations with Different Evolution Behaviors. In *IEEE 1996 Int. Conf. on Evolutionary Computation*, pages 188–191, 1996.
- [6] R.M. Aiex, S.L. Martins, C.C. Ribeiro, and N.R. Rodriguez. Asynchronous Parallel Strategies for Tabu Search Applied to the Partitioning of VLSI Circuits. *Monografias em ciência da computação*, Pontifícia Universidade Católica de Rio de Janeiro, 1996.
- [7] J.R.A. Allwright and D.B. Carpenter. A Distributed Implementation of Simulated Annealing for the Travelling Salesman Problem. *Parallel Computing*, 10:335–338, 1989.
- [8] A.A. Andreatta and Ribeiro C.C. A Graph Partitioning Heuristic for the Parallel Pseudo-Exhaustive Logical Test of VLSI Combinational Circuits. *Annals of Operations Research*, 50:1–36, 1994.
- [9] R. Azencott. Parallel Simulated Annealing: An Overview of Basic Techniques. In R. Azencott, editor, *Simulated Annealing Parallelization Techniques*, pages 37–46. John Wiley & Sons, New York, NY, 1992.
- [10] R. Azencott, editor. *Simulated Annealing Parallelization Techniques*. John Wiley & Sons, New York, NY, 1992a.
- [11] P. Badeau, F. Guertin, M. Gendreau, J.-Y. Potvin, and É.D. Taillard. A Parallel Tabu Search Heuristic for the Vehicle Routing Problem with Time Windows. *Transportation Research C*, 5(2):109–122, 1997.
- [12] F. Baiardi and S. Orlando. Strategies for a Massively Parallel Implementation of Simulated Annealing. In *Parallel Architectures and Languages, PARLE'89*, pages 273–287, 1989.

- [13] S. Baluja. A Massively Distributed Parallel Genetic Algorithm. Technical Report CMU-CS-92-196R, Carnegie Mellon University, 1992.
- [14] S. Baluja. Structure and Performance of Fine-Grain Parallelism in Genetic Algorithms. In S. Forrest, editor, *Proceedings of the Fifth International Conference on Genetic Algorithms*, pages 155–162. Morgan Kaufmann, San Mateo, CA, 1993.
- [15] P. Banerjee and M. Jones. A Parallel Simulated Annealing Algorithm for Standard Cell Placement on Hypercube Computer. In *Proceedings of the IEEE International Conference on Computer-Aided Design: ICCAD-86*, pages 34–37. IEEE Computer Society Press, Washington, DC, 1986.
- [16] P. Banerjee, M. Jones, and J. Sargent. Parallel Simulated Annealing Algorithm for Cell Placement on Hypercube Multiprocessors. *IEEE Transactions on Parallel and Distributed Systems*, 1(1):91–106, 1990.
- [17] J. Bannister and M. Gerla. Design of the Wavelength-Division Optical Network. Report CSD-890022, UCLA Computer Science Department, 1989.
- [18] V.C. Barbosa and E. Gafni. A Distributed Implementation of Simulated Annealing. *Journal of Parallel and Distributed Computing*, 6(2):411–434, 1989.
- [19] R.S. Barr, B.L. Golden, J.P. Kelly, M.G.C. Resende, and W.R. Stewart. Designing and Reporting on Computational Experiments with Heuristic Methods. *Journal of Heuristics*, 1(1):9–32, 1995.
- [20] R.S. Barr and B.L. Hickman. Reporting Computational Experiments with Parallel Algorithms: Issues, Measures, and Experts Opinions. *ORSA Journal on Computing*, 5(1):2–18, 1993.
- [21] R. Battiti and G. Tecchioli. Parallel Based Search for Combinatorial Optimization: Genetic Algorithms and TABU. *Microprocessors and Microsystems*, 16(7):351–367, 1992.
- [22] D. Beasley, D.R. Bull, and R.R. Martin. An Overview of Genetic Algorithms: Part 1, Fundamentals. *University Computing*, 15(2):58–69, 1993.
- [23] T. Belding. The Distributed Genetic Algorithm Revisited. In L.J. Eshelman, editor, *Proceedings of the Sixth International Conference on Genetic Algorithms*, pages 114–121. Morgan Kaufmann, San Mateo, CA, 1995.
- [24] R. Bianchini and C.M. Brown. Parallel Genetic Algorithms on Distributed-Memory Architectures. Technical Report TR 436, University of Rochester, Computer Science Department, 1992.
- [25] N. Boissin and J.L. Lutton. A Parallel Simulated Annealing Algorithm. *Parallel Computing*, 19(8):859–872, 1993.
- [26] G. Bongiovanni, P. Crescenzi, and C. Guerra. Parallel Simulated Annealing for Shape Detection. *Computer Vision and Image Understanding*, 61(1):60–69, 1995.

- [27] R. Borut and J. Silc. Using Parallel Simulated Annealing in the Mapping Problem. *Lecture Notes in Computer Science 817*, pages 797–800, 1994.
- [28] T. Boseniuk and W. Ebeling. Optimization of NP-Complete Problems by Boltzmann-Darwin Strategies Including Life Cycles. *Europhysics Letters*, 6(2):107–112, 1988.
- [29] T. Boseniuk and W. Ebeling. Boltzmann, Darwin and Haeckel Strategies in Optimization Problems. In H.-P. Schwefel and R. Männer, editors, *Parallel Problem Solving from Nature, Lecture Notes in Computer Science 496*, pages 430–444. Springer-Verlag, Berlin, 1991.
- [30] T. Boseniuk, W. Ebeling, and A. Engel. Boltzmann and Darwin Strategies in Complex Optimization. *Physics Letters A*, 125(6/7):307–310, 1987.
- [31] J. Branke, U. Kohlmorgen, and H. Schmeck. A Distributed Genetic Algorithm Improving the Generalization Behavior of Neural Networks. *Lecture Notes in Computer Science 912*, pages 107–121, 1995.
- [32] R. Brouwer and P. Banerjee. A Parallel Simulated Annealing Algorithm for Channel Routing on a Hypercube Multiprocessor. In *Proceedings of the IEEE International Conference on Computer-Aided Design: ICCAD-88*, pages 4–7. IEEE Computer Society Press, Washington, DC, 1988.
- [33] D.E. Brown, C.L. Huntley, and A.R. Spillane. A parallel genetic heuristic for the quadratic assignment problem. In J.D. Schaffer, editor, *Proceedings of the Third International Conference on Genetic Algorithms*, pages 406–415. Morgan Kaufmann, San Mateo, CA, 1989.
- [34] E. Cantú-Paz. A Summary of Research on Parallel Genetic Algorithms. Report 95007, University of Illinois at Urbana-Champaign, 1995.
- [35] E. Cantú-Paz and D.E. Goldberg. Predicting Speedups of Ideal Bounding Cases of Parallel Genetic Algorithms. Report 96008, University of Illinois at Urbana-Champaign, 1996.
- [36] A. Casotto, F. Romeo, and A.L. Sangiovanni-Vincentelli. A Parallel Simulated Annealing Algorithm for the Placement of Macro-Cells. In *Proceedings of the IEEE International Conference on Computer-Aided Design: ICCAD-86*, pages 30–33. IEEE Computer Society Press, Washington, DC, 1986.
- [37] A. Casotto, F. Romeo, and A.L. Sangiovanni-Vincentelli. A Parallel Simulated Annealing Algorithm for the Placement of Macro-Cells. *IEEE Transactions on Computer-Aided Design*, 6(5):838–847, 1987.
- [38] A. Casotto, F. Romeo, and A.L. Sangiovanni-Vincentelli. Placement of Standard Cells Using Simulated Annealing on the Connection Machine. In *Proceedings of the IEEE International Conference on Computer-Aided Design: ICCAD-87*, pages 350–353. IEEE Computer Society Press, Washington, DC, 1987a.

- [39] J. Chakrapani and J. Skorin-Kapov. A Connectionist Approach to the Quadratic Assignment Problem. *Computers & Operations Research*, 19(3/4):287–295, 1992.
- [40] J. Chakrapani and J. Skorin-Kapov. Massively Parallel Tabu Search for the Quadratic Assignment Problem. *Annals of Operations Research*, 41:327–341, 1993.
- [41] J. Chakrapani and J. Skorin-Kapov. Connection Machine Implementation of a Tabu Search Algorithm for the Traveling Salesman Problem. *Journal of Computing and Information Technology*, 1(1):29–36, 1993a.
- [42] J. Chakrapani and J. Skorin-Kapov. Mapping Tasks to Processors to Minimize Communication Time in a Multiprocessor System. In *The Impact of Emerging Technologies of Computer Science and Operations Research*, pages 45–64. Kluwer, Norwell, MA, 1995.
- [43] R.D. Chamberlain, M.N. Edelman, M.A. Franklin, and E.E. Witte. Simulated Annealing on a Multiprocessor. In *Proceedings of the IEEE International Conference on Computer-Aided Design: ICCAD-88*, pages 540–544. IEEE Computer Society Press, Washington, DC, 1988.
- [44] H. Chen and N.S. Flann. Parallel Simulated Annealing and Genetic Algorithms: A Space of Hybrid Methods. In Y. Davidor, H.-P. Schwefel, and R. Männer, editors, *Parallel Problem Solving from Nature III, Lecture Notes in Computer Science 866*, pages 428–438. Springer-Verlag, Berlin, 1994.
- [45] R.J. Chen, R. Meyer, and J. Yeckel. A Genetic Algorithm for Diversity Minimization and its Parallel Implementation. In S. Forrest, editor, *Proceedings of the Fifth International Conference on Genetic Algorithms*, pages 163–170. Morgan Kaufmann, San Mateo, CA, 1993.
- [46] Y.-W. Chen, Z. Nakao, and X. Fang. Parallelization of a Genetic Algorithm for Image Restoration and its Performance Analysis. In *IEEE International Conference on Evolutionary Computation*, pages 463–468, 1996.
- [47] N. Christofides, Mingozzi A., and P. Toth. The Vehicle Routing Problem. In N. Christofides, Mingozzi A., P. Toth, and C. Sandi, editors, *Combinatorial Optimization*, pages 315–338. John Wiley, New York, 1979.
- [48] M.J. Chung and K.K. Rao. Parallel Simulated Annealing for Partitioning and Routing. In *Proceedings of the IEEE International Conference on Computer Design: VLSI in Computers; ICCD'86*, pages 238–242. IEEE Computer Society Press, Washington, DC, 1986.
- [49] J. Cohoon, S. Hedge, W. Martin, and D. Richards. Punctuated Equilibria: A Parallel Genetic Algorithm. In J.J. Grefenstette, editor, *Proceeding of the Second International Conference on Genetic Algorithms and their Applications*, pages 148–154. Lawrence Erlbaum Associates, Hillsdale, NJ, 1987.

- [50] J. Cohoon, S. Hedge, W. Martin, and D. Richards. Distributed Genetic Algorithms for the Floorplan Design Problem. *IEEE Transactions on Computer-Aided Design*, 10:483–492, 1991c.
- [51] J. Cohoon, W. Martin, and D. Richards. Genetic Algorithm and Punctuated Equilibria in VLSI. In H.-P. Schwefel and R. Männer, editors, *Parallel Problem Solving from Nature, Lecture Notes in Computer Science 496*, pages 134–144. Springer-Verlag, Berlin, 1991a.
- [52] J. Cohoon, W. Martin, and D. Richards. A Multi-Population Genetic Algorithm for Solving the k-Partition Problem on Hyper-Cubes. In R.K. Belew and L.B. Booker, editors, *Proceedings of the Fourth International Conference on Genetic Algorithms*, pages 134–144. Morgan Kaufmann, San Mateo, CA, 1991b.
- [53] R.J. Collins and D.R. Jefferson. Selection in Massively Parallel Genetic Algorithms. In R.K. Belew and L.B. Booker, editors, *Proceedings of the Fourth International Conference on Genetic Algorithms*, pages 249–256. Morgan Kaufmann, San Mateo, CA, 1991.
- [54] A. Colorni, M. Dorigo, and V. Manniezzo. Distributed Optimization by Ant Colonies. In *Proceedings European Conference on Artificial Life*, pages 134–142. North-Holland, Amsterdam, 1991.
- [55] A. Colorni, M. Dorigo, and V. Manniezzo. An Investigation of Some Properties of an ‘Ant Algorithm’. In R. Männer and B. Manderick, editors, *Parallel Problem Solving from Nature II*, pages 509–520. North-Holland, Amsterdam, 1992.
- [56] T.G. Crainic and M. Gendreau. A Cooperative Parallel Tabu Search for Capacitated Network Design. Publication, Centre de recherche sur les transports, Université de Montréal, Montréal, QC, Canada, 1997.
- [57] T.G. Crainic and M. Gendreau. Towards an Evolutionary Method - Cooperating Multi-Thread Parallel Tabu Search Hybrid. Publication CRT-97-27, Centre de recherche sur les transports, Université de Montréal, Montréal, QC, Canada, 1997.
- [58] T.G. Crainic and G. Laporte. Planning Models for Freight Transportation. *European Journal of Operational Research*, pages 409–438, 1997.
- [59] T.G. Crainic, M. Toulouse, and M. Gendreau. Synchronous Tabu Search Parallelization Strategies for Multicommodity Location-Allocation with Balancing Requirements. *OR Spektrum*, 17(2/3):113–123, 1995.
- [60] T.G. Crainic, M. Toulouse, and M. Gendreau. Parallel Asynchronous Tabu Search for Multicommodity Location-Allocation with Balancing Requirements. *Annals of Operations Research*, 63:277–299, 1995a.
- [61] T.G. Crainic, M. Toulouse, and M. Gendreau. Towards a Taxonomy of Parallel Tabu Search Algorithms. *INFORMS Journal on Computing*, 9(1):61–72, 1997.

- [62] J. Cui and T.C. Fogarty. Optimization by Using a Parallel Genetic Algorithm on a Transputer Computing Surface. In M. Valero, E. Onate, M. Jane, J.L. Larriba, and B. Suarez, editors, *Parallel Computing and Transputer Applications*, pages 246–254. IOS Press, Amsterdam, 1992.
- [63] F. Darema, S. Kirkpatrick, and V.A. Norton. Parallel Algorithms for Chip Placement by Simulated Annealing. *IBM Journal of Research and Development*, 31:391–402, 1987.
- [64] F. Darema and G. F. Pfister. Multipurpose Parallelism in VLSI Computer-Aided Design: A Case Study. Report Research Report RC12516, IBM T.J. Watson Research Center, 1987a.
- [65] M. Davis, L. Liu, and J.G. Elias. VLSI Circuit Synthesis Using a Parallel Genetic Algorithm. In *IEEE'94 Conference on Evolutionary Computing*, pages 104–109, 1994.
- [66] I. De Falco, R. Del Balio, and E. Tarantino. Solving the Mapping Problem by Parallel Tabu Search. Report, Istituto per la Ricerca sui Sistemi Informatici Paralleli-CNR, 1995.
- [67] I. De Falco, R. Del Balio, E. Tarantino, and R. Vaccaro. Improving Search by Incorporating Evolution Principles in Parallel Tabu Search. In *Proc. Int. Conf. on Machine Learning*, pages 823–828, 1994.
- [68] K.A. De Jong. *An Analysis of the Behavior of a Class of Genetic Adaptive Systems*. PhD thesis, University of Michigan, 1975.
- [69] S. Devadas and A.R. Newton. Topological Optimization of Multiple Level Array Logic: on Uni and Multi-Processors. In *Proceedings of the IEEE International Conference on Computer-Aided Design: ICCAD-86*, pages 38–41. IEEE Computer Society Press, Washington, DC, 1986.
- [70] R. Diekmann, R. Lüling, and J. Simon. Problem Independent Distributed Simulated Annealing and its Applications. In R.V.V. Vidal, editor, *Lecture Notes in Economics and Mathematical Systems, volume 396*, pages 17–44. Springer Verlag, Berlin, 1993.
- [71] M. Dorigo. *Ottimizzazione, Apprendimento Automatico, ed Algoritmi Basati su Metafora Naturale*. PhD thesis, Politecnico di Milano, 1992.
- [72] M. Dorigo, V. Manniezzo, and A. Colorni. The Ant System: Optimization by a Colony of Cooperating Agents. *IEEE Transactions on Systems, Man, and Cybernetics - Part B*, 26(1):29–41, 1996.
- [73] M.D. Durand. Parallel Simulated Annealing: Accuracy vs. Speed in Placement. *IEEE Design & Test of Computers*, 6(3):8–34, 1989.
- [74] M.D. Durand. Cost Function Error in Asynchronous Parallel Simulated Annealing Algorithms. Technical Report CUCS-423-89, University of Columbia, 1989a.

- [75] S. Elo. A Parallel Genetic Algorithm on the CM-2 for Multi-Modal Optimization. In *Proc. Int. Conf. on Machine Learning*, pages 818–822, 1994.
- [76] E. Felten, S. Karlin, and S.W. Otto. The Traveling Salesman Problem on a Hypercube, MIMD Computer. In *Proc. 1985 of the Int. Conf. on Parallel Processing*, pages 6–10, 1985.
- [77] T.A. Feo and M.G.C. Resende. Greedy Randomized Adaptive Search Procedures. *Journal of Global Optimization*, 6(2):109–133, 1995.
- [78] T.A. Feo, M.G.C. Resende, and S.H. Smith. A Greedy Randomized Adaptive Search Procedure for Maximum Independent Set. *Operations Research*, 42(5):860–878, 1994.
- [79] C.-N. Fiechter. A parallel tabu search algorithm for large travelling salesman problems. *Discrete Applied Mathematics*, 51(3):243–267, 1994.
- [80] M.A. Fleischer. Cybernetic Optimization by Simulated Annealing: Accelerating Convergence by Parallel Processing and Probabilistic Feedback Control. *Journal of Heuristics*, 1(2):225–246, 1996.
- [81] M.A. Fleischer and S.H. Jacobson. Optimization by Simulated Annealing: An Implementation of Parallel Processing Using Probabilistic Feedback Control. In I.H. Osman and J.P. Kelly, editors, *Meta-Heuristics: Theory & Applications*, pages 249–275. Kluwer, Norwell, MA, 1996.
- [82] T.C. Fogarty and R. Huang. Implementing the Genetic Algorithm on Transputer Based Parallel Systems. In H.-P. Schwefel and R. Männer, editors, *Proceedings of the 1st Workshop on Parallel Problem Solving from Nature*, pages 145–149. Springer-Verlag, Berlin, 1990.
- [83] D.B. Fogel. Evolutionary Programming: An Introduction and Some Current Directions. *Statistics and Computing*, 4:113–130, 1994.
- [84] S. Forrest. Emergent Computation: Self-Organizing, Collective, and Cooperative Phenomena in Natural and Artificial Computing Network. In S. Forrest, editor, *Emergent Computation*, pages 1–11. MIT/North-Holland, 1991.
- [85] B.L. Garcia, J.-Y. Potvin, and J.M. Rousseau. A Parallel Implementation of the Tabu Search Heuristic for Vehicle Routing Problems with Time Window Constraints. *Computers & Operations Research*, 21(9):1025–1033, 1994.
- [86] M. Gendreau, P. Badeau, F. Guertin, J.-Y. Potvin, and É.D. Taillard. A Solution Procedure for Real-Time Routing and Dispatching of Commercial Vehicles. Publication CRT-96-24, Centre de recherche sur les transports, Université de Montréal, 1996.
- [87] B. Gendron and T.G. Crainic. Parallel Branch-and-Bound Algorithms: Survey and Synthesis. *Operations Research*, 42(6):1042–1066, 1994.

- [88] F. Ghannadian, R. Shonkwiler, and C.O. Alford. Parallel Simulated Annealing for the n -Queen Problem. In *IPPS: 7th International Parallel Processing Symposium*, pages 690–694. IEEE Computer Society Press, 1993.
- [89] A. Giordana, L. Saitta, and F. Zini. Learning Disjunctive Concepts by Means of Genetic Algorithms. In *Proc. Int. Conf. on Machine Learning*, pages 96–104, 1994.
- [90] K.A. Girodias, H.H. Barrett, and R.L. Shoemaker. Parallel Simulated Annealing Algorithm for Emission Tomography. *Phys. Med. Biol.*, 36(7):921–938, 1991.
- [91] F. Glover. Future Paths for Integer Programming and Links to Artificial Intelligence. *Computers & Operations Research*, 1(3):533–549, 1986.
- [92] F. Glover. Tabu Search – Part I. *ORSA Journal on Computing*, 1(3):190–206, 1989.
- [93] F. Glover. Tabu Search – Part II. *ORSA Journal on Computing*, 2(1):4–32, 1990.
- [94] F. Glover. Tabu Search and Adaptive Memory Programming – Advances, Applications and Challenges. In R.S. Barr, R.V. Helgason, and J. Kennington, editors, *Interfaces in Computer Science and Operations Research*, pages 1–75. Kluwer, Norwell, MA, 1996.
- [95] F. Glover and M. Laguna. Tabu search. In C.R. Reeves, editor, *Modern Heuristic Techniques for Combinatorial Problems*, pages 70–150. Blackwell Scientific Publications, Oxford, 1993.
- [96] D.E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley, Reading, MA, 1989.
- [97] D.E. Goldberg. A Note on Boltzmann Tournament Selection for Genetic Algorithms and Population-Oriented Simulated Annealing. *Complex Systems*, 4:445–460, 1990.
- [98] D.E. Goldberg and K. Deb. A Comparative Analysis of Selection Schemes Used in Genetic Algorithms. In G.J.E. Rawlins, editor, *Foundations of Genetic Algorithm & Classifier Systems*, pages 69–93. Morgan Kaufman, San Mateo, CA, 1991.
- [99] D.E. Goldberg and J. Richardson. Genetic Algorithms with Sharing for Multimodal Function Optimization. In J.J. Grefenstette, editor, *Proceeding of the Second International Conference on Genetic Algorithms and their Applications*, pages 41–49. Lawrence Erlbaum Associates, Hillsdale, NJ, 1987.
- [100] B.L. Golden, E.A. Wasil, J.P. Kelly, and I.-M. Chao. The Impact of Metaheuristics on Solving the Vehicle Routing Problem: Algorithms, Problem Sets, and Computational Results. In T.G. Crainic and G. Laporte, editors, *Fleet Management and Logistics*. Kluwer, Norwell, MA, 1998.
- [101] V. Gordon and D. Whitley. Serial and Parallel Genetic Algorithms as Function Optimizers. In S. Forrest, editor, *Proceedings of the Fifth International Conference on Genetic Algorithms*, pages 177–183. Morgan Kaufmann, San Mateo, CA, 1993.

- [102] V. Gordon, D. Whitley, and A. Bohm. Dataflow Parallelism in Genetic Algorithms. In R. Männer and B. Manderick, editors, *Parallel Problem Solving from Nature II*, pages 533–542. North-Holland, Amsterdam, 1992.
- [103] V.S. Gordon. Locality in Genetic Algorithms. In *Proceedings International Conference on Machine Learning*, pages 428–432, 1994.
- [104] V.S. Gordon and D. Whitley. A Machine-Independent Analysis of Parallel Genetic Algorithms. *Complex Systems*, 8:181–214, 1994.
- [105] M. Gorges-Schleuter. ASPARAGOS: An Asynchronous Parallel Genetic Optimization Strategy. In J.D. Schaffer, editor, *Proceedings of the Third International Conference on Genetic Algorithms*, pages 422–427. Morgan Kaufmann, San Mateo, CA, 1989.
- [106] M. Gorges-Schleuter. ASPARAGOS: A Parallel Genetic Algorithm for Population Genetics. In J.D. Becker, I. Eisele, and F.W. Mundemann, editors, *Parallelism, Learning, Evolution. Workshop on Evolutionary Models and Strategies - WOPLOT 89*, pages 407–418. Springer-Verlag, Berlin, 1991.
- [107] M. Gorges-Schleuter. Explicit Parallelism of Genetic Algorithm Through Population Structures. In H.-P. Schwefel and R. Männer, editors, *Parallel Problem Solving from Nature, Lecture Notes in Computer Science 496*, pages 150–160. Springer-Verlag, Berlin, 1991a.
- [108] M. Gorges-Schleuter. Comparison of Local Mating Strategies in Massively Parallel Genetic Algorithms. In R. Männer and B. Manderick, editors, *Parallel Problem Solving from Nature II*, pages 553–562. North-Holland, Amsterdam, 1992.
- [109] C. Graffigne. Parallel Annealing by Periodically Interacting Multiple Searches: an Experimental Study. In R. Azencott, editor, *Simulated Annealing Parallelization Techniques*, pages 47–79. John Wiley & Sons, New York, NY, 1992.
- [110] P. Graham and B. Nelson. A Hardware Genetic Algorithm for the Travelling Salesman Problem on SPLASH 2. In W. Moore and W. Luk, editors, *Field-Programmable Logic and Applications*, pages 352–361. Springer Verlag, Oxford, England, 1995.
- [111] D.R. Greening. A Taxonomy of Parallel Simulated Annealing Techniques. Technical Report No. RC 14884, IBM, 1989a.
- [112] D.R. Greening. Asynchronous Parallel Simulated Annealing. *Lectures in Complex Systems*, 3:497–505, 1990.
- [113] D.R. Greening. Parallel Simulated Annealing Techniques. *Physica D*, 42:293–306, 1990.
- [114] D.R. Greening and F. Darema. Rectangular Spatial Decomposition Methods for Parallel Simulated Annealing. In *Proceedings of the International Conference on Supercomputing*, pages 295–302, 1989.

- [115] J. Grefenstette. Parallel Adaptive Algorithms for Function Optimization. Technical Report CS-81-19, Vanderbilt University, Nashville, 1981.
- [116] P. Grosso. *Computer Simulations of Genetic Adaptation: Parallel Subcomponent Interaction in a Multilocus Model*. PhD thesis, University of Michigan, 1985.
- [117] R. Hauser and R. Männer. Implementation of Standard Genetic Algorithm on MIMD Machines. In Y. Davidor, H.-P. Schwefel, and R. Männer, editors, *Parallel Problem Solving from Nature III, Lecture Notes in Computer Science 866*, pages 504–514. Springer-Verlag, Berlin, 1994.
- [118] M. Herdy. Reproductive Isolation as Strategy Parameter in Hierarchical Organized Evolution Strategies. In R. Männer and B. Manderick, editors, *Parallel Problem Solving from Nature II*, pages 207–217. North-Holland, Amsterdam, 1992.
- [119] F. Hoffmeister. Scalable Parallelism by Evolutionary Algorithms. In M. Grauer and D.B. Pressmar, editors, *Lecture Notes in Mathematical Systems and Economics*. Springer-Verlag, Berlin, 1991.
- [120] T. Hogg and C. Williams. Solving the Really Hard Problems with Cooperative Search. In *Proc. of the 11th Natl. Conf. on Artificial Intelligence (AAAI93)*, pages 231–236. AAAI Press, 1993.
- [121] J.H. Holland. *Adaptation in Natural and Artificial Systems*. University of Michigan Press, Ann Arbor, MI, 1975.
- [122] J.H. Holland. Royal Road Functions. *Genetic Algorithm Digest*, 7(22), 1993.
- [123] K. Holmqvist, A. Migdalas, and P.M. Pardalos. Parallelized Heuristics for Combinatorial Search. In A. Migdalas, P.M. Pardalos, and S. Storoy, editors, *Parallel Computing in Optimization*, pages 269–294. Kluwer, Norwell, MA, 1997.
- [124] C-E. Hong and B. M. McMillin. Relaxing Synchronization in Distributed Simulated Annealing. Report C.Sc. 92-06, University of Missouri at Rolla, 1992.
- [125] C.L. Huntley and D.E. Brown. Parallel Genetic Algorithms with Local Search. *Computers & Operations Research*, 23(6):559–571, 1996.
- [126] P. Husbands and F. Mill. Simulated Co-Evolution as the Mechanism for Emergent Planning and Scheduling. In R.K. Belew and L.B. Booker, editors, *Proceedings of the Fourth International Conference on Genetic Algorithms*, pages 264–270. Morgan Kaufmann, San Mateo, CA, 1991.
- [127] R. Jayaraman and F. Darena. Error Tolerance in Parallel Simulated Techniques. In *Proceedings of the IEEE International Conference on Computer-Aided Design: ICCAD-88*, pages 545–548. IEEE Computer Society Press, Washington, DC, 1988.
- [128] R. Jayaraman and Rutenbar R.A. Floorplanning by Annealing on Hypercube Multiprocessor. In *Proceedings of the IEEE International Conference on Computer-Aided Design: ICCAD-87*, pages 346–349. IEEE Computer Society Press, Washington, DC, 1987.

- [129] C. Jayewardena. Parallel Simulated Annealing: Image Reconstruction on a Network of Transputers. Technical Report 90-014, University of Bern, 1990.
- [130] C.-S. Jeong and M.-H. Kim. Parallel Algorithm for the TSP on SIMD Machines Using Simulated Annealing. In *Proceedings of the International Conference on Application Specific Array Processors*, pages 712–721, 1990.
- [131] C.-S. Jeong and M.-H. Kim. Fast Parallel Simulated Annealing Algorithm for TSP on SIMD Machines with Linear Interconnections. *Parallel Computing*, 17:221–228, 1991.
- [132] P. Jog. *Parallelization of Probabilistic Sequential Search Algorithms*. PhD thesis, Indiana University, Bloomington, 1989.
- [133] P. Jog and D Gucht. Parallelization of Probabilistic Sequential Search Algorithms. In J.J. Grefenstette, editor, *Proceeding of the Second International Conference on Genetic Algorithms and their Applications*, pages 170–176. Lawrence Erlbaum Associates, Hillsdale, NJ, 1987.
- [134] P. Jog, J.Y. Suh, and D.V. Gucht. Parallel Genetic Algorithms Applied to the Traveling Salesman Problem. *SIAM Journal of Optimization*, 1(4):515–529, 1991.
- [135] M. Jones and P. Banerjee. Performance of a Parallel Algorithm for Standard Cell Placement on the Intel Hypercube. In *Proceedings of the 24th Design Automation and Machine Conference*, pages 768–778, 1987.
- [136] J.O. Kephart, T. Hogg, and B.A. Huberman. Dynamics of Computational Ecosystems: Implication for DAI. *Distributed Artificial Intelligence*, 2, 1989.
- [137] J.O. Kephart, T. Hogg, and B.A. Huberman. Collective behavior of predictive agents. *Physica D*, 42:48–65, 1990.
- [138] Y. Kim and M. Kim. A Step-Wise-Overlapped Parallel Annealing Algorithm on a Message-Passing Multiprocessor System. *Concurrency: Practice & Experience*, 2(2):123–148, 1990.
- [139] S. Kirkpatrick, C.D. Gelatt, and M.P. Vecchi. Optimization by Simulated Annealing. *Science*, 220:671–680, 1983.
- [140] R. L. Knight and R.L. Wainwright. HYPERGEN” - A Distributed Genetic Algorithm on a Hypercube. In *Proceedings of the 1992 IEEE Scalable High Performance Computing Conference*, pages 232–235. IEEE Computer Society Press, Los Alamitos, CA, 1992.
- [141] V. Kommu and I. Pomeranz. Effect of Communication in a Parallel Genetic Algorithm. In *Proceedings of the 1992 International Conference on Parallel Processing*, volume III, Algorithms and Applications, pages 310–317. CRC Press, 1992.

- [142] C. Kosak, J. Marks, and S. Shieber. A Parallel Genetic Algorithm for Network-Diagram Layout. In R.K. Belew and L.B. Booker, editors, *Proceedings of the Fourth International Conference on Genetic Algorithms*, pages 458–465. Morgan Kaufmann, San Mateo, CA, 1991.
- [143] P. Kraft, M. Nölle, Schreiber. G., S. Marshall, and H. Hans Burkhardt. A Parallel Genetic Algorithm for Optimizing Morphological Filters on Inhomogeneous Workstation Clusters. In S. Miguet and A. Montanvert, editors, *Proceedings of the Fourth International Workshop on Parallel Image Analysis (IWPIA '95)*, pages 171–182. LIP-ENS, Lyon, France, 1995.
- [144] R. Kravitz and R. Rutenbar. Multiprocessor-Based Placement by Simulated Annealing. In *Proceedings of the 23th ACM/IEEE Design Automation Conference*, pages 567–573, 1986.
- [145] S.A. Kravitz and R. Rutenbar. Placement by Simulated Annealing on a Multiprocessor. *IEEE Transactions on Computer-Aided Design*, 6:534–549, 1987.
- [146] B. Kröger, P. Schwenderling, and O. Vomberger. Parallel Genetic Packing of Rectangles. In H.-P. Schwefel and R. Männer, editors, *Parallel Problem Solving from Nature, Lecture Notes in Computer Science 496*, pages 160–164. Springer-Verlag, Berlin, 1991.
- [147] P. Laarhoven and E. Aarts. *Simulated Annealing: Theory and Applications*. Reidel, Dordrecht, 1987.
- [148] G. von Laszewski and H. Mühlenbein. Partitioning a Graph with a Parallel Genetic Algorithm. In H.-P. Schwefel and R. Männer, editors, *Parallel Problem Solving from Nature, of Lecture Notes in Computer Science 496*, pages 165–169. Springer-Verlag, Berlin, 1991.
- [149] P.S. Laursen. Problem-Independent Parallel Simulated Annealing Using Selection and Migration. In Y. Davidor, H.-P. Schwefel, and R. Männer, editors, *Parallel Problem Solving from Nature III, Lecture Notes in Computer Science 866*, pages 408–417. Springer-Verlag, Berlin, 1994.
- [150] P.S. Laursen. Parallel Heuristic Search – Introductions and a New Approach. In A. Ferreira and P.M. Pardalos, editors, *Solving Combinatorial Optimization Problems in Parallel, Lecture Notes in Computer Science 1054*, pages 248–274. Springer-Verlag, Berlin, 1996.
- [151] C.Y.. Lee and S.J. Kim. Parallel Genetic Algorithms for the Earliness-Tardiness Job Scheduling Problem with General Penalty Weights. *Computers & Industrial Engineering*, 28:231–243, 1995.
- [152] F.-H. A. Lee. *Parallel Simulated Annealing on a Message-Passing Multi-Computer*. PhD thesis, Utah State University, 1995.
- [153] K-G. Lee and S-Y. Lee. Efficient Parallelization of Simulated Annealing Using Multiple Markov Chains: An Application to Graph Partitioning. In T.N. Mudge, editor,

- Proceedings of the International Conference on Parallel Processing*, pages 177–180. CRC Press, 1992a.
- [154] S.-Y. Lee and K.-G. Lee. Asynchronous Communication of Multiple Markov Chains in Parallel Simulated Annealing. In T.N. Mudge, editor, *Proceedings of the International Conference on Parallel Processing*, volume III: Algorithms and Applications, pages 169–176. CRC Press, Boca Raton, FL, 1992b.
- [155] D. Levine. A parallel genetic algorithm for the set partitioning problem. In I.H. Osman and J.P. Kelly, editors, *Meta-Heuristics: Theory & Applications*, pages 23–35. Kluwer, Norwell, MA, 1996.
- [156] F.-T. Lin, C.-Y. Kao, and C.-C. Hsu. Incorporating Genetic Algorithms into Simulated Annealing. In *Proceedings of the Fourth Int. Symp. on AI*, pages 290–297, 1991.
- [157] S.-C. Lin, W. Punch, and E. Goodman. Coarse-Grain Parallel Genetic Algorithms: Categorization and New Approach. In *Sixth IEEE Symposium on Parallel and Distributed Processing*, pages 28–37. IEEE Computer Society Press, 1994.
- [158] J. Lis. Parallel Genetic Algorithm with the Dynamic Control Parameter. In *IEEE 1996 Int. Conf. on Evolutionary Computation*, pages 324–328, 1996.
- [159] S. W. Mahfoud and D. E. Goldberg. Parallel Recombinative Simulated Annealing: A Genetic Algorithm. *Parallel Computing*, 21:1–28, 1995.
- [160] S.W. Mahfoud. A Comparison of Parallel and Sequential Niching Methods. In L.J. Eshelman, editor, *Proceedings of the Sixth International Conference on Genetic Algorithms*, pages 136–143. Morgan Kaufmann, San Mateo, CA, 1995.
- [161] M. Malek, M Guruswamy, M. Pandya, and H. Owens. Serial and Parallel Simulated Annealing and Tabu Search Algorithms for the Traveling Salesman Problem. *Annals of Operations Research*, 21:59–84, 1989.
- [162] B. Manderick and P. Spiessens. Fine-Grained Parallel Genetic Algorithm. In J.D. Schaffer, editor, *Proceedings of the Third International Conference on Genetic Algorithms*, pages 416–421. Morgan Kaufmann, San Mateo, CA, 1989.
- [163] V. Maniezzo. Genetic Evolution of the Topology and Weight Distribution of Neural Networks. *IEEE Transactions on Neural Networks*, 5(1):39–53, 1993.
- [164] S.L. Martins, C.C. Ribeiro, and N.R. Rodriguez. Parallel Programming Tools for Distributed Memory Environments. Monografias em Ciência da Computação 01/96, Pontifícia Universidade Católica de Rio de Janeiro, 1996.
- [165] T. Maruyama, T. Hirose, and A. Konagaya. A Fine-Grained Parallel Genetic Algorithm for Distributed Parallel Systems. In S. Forrest, editor, *Proceedings of the Fifth International Conference on Genetic Algorithms*, pages 184–190. Morgan Kaufmann, San Mateo, CA, 1993.

- [166] T. Maruyama, A. Konagaya, and K. Konishi. An Asynchronous Fine-Grained Parallel Genetic Algorithm. In R. Männer and B. Manderick, editors, *Parallel Problem Solving from Nature II*, pages 563–572. North-Holland, Amsterdam, 1992.
- [167] C. Mazza. Parallel simulated annealing. *RSA: Random Structures & Algorithms*, 3(2):139–148, 1992.
- [168] L.D. Merkle and G.B. Lamont. Comparison of Parallel Messy Genetic Algorithms Data Distribution Strategies. In S. Forrest, editor, *Proceedings of the Fifth International Conference on Genetic Algorithms*, pages 191–198. Morgan Kaufmann, San Mateo, CA, 1993.
- [169] N. Metropolis, A. Rosenbluth, M. Rosenbluth, A. Teller, and E. Teller. Equation of State Calculation by Fast Computing Machines. *Journal of Chemical Physics*, 21:1087–1092, 1953.
- [170] Z. Michalewicz. *Genetic Algorithms + Data Structures = Evolution Programs*. Springer-Verlag, Berlin, 1992.
- [171] K.R. Miller and R. Shonkwiler. Genetic Algorithm/Neural Network Synergy for Non-linearly Constrained Optimization Problems. In *Combinations of Genetic Algorithms and Neural Networks*, pages 248–257. IEEE Computer Society, 1992.
- [172] G. Monien, F. Ramme, and H. Salmen. A Parallel Simulated Annealing Algorithm for Generating 3D Layouts of Undirected Graphs. *Lecture Notes in Computer Science 1027*, pages 396–408, 1995.
- [173] P. Moscato. On Evolution, Search, Optimization, Genetic Algorithms and Martial Arts: Towards Memetic Algorithms. Publication Report 790, Caltech Concurrent Computation Program, 1989.
- [174] P. Moscato and M.G. Norman. A ‘Memetic’ Approach for the Traveling Salesman Problem. Implementation of a Computational Ecology for Combinatorial Optimization on Message-Passing Systems. In M. Valero, E. Onate, M. Jane, J.L. Larriba, and B. Suarez, editors, *Parallel Computing and Transputer Applications*, pages 187–194. IOS Press, Amsterdam, 1992.
- [175] H. Mühlenbein. Parallel Genetic Algorithms, Population Genetics, and Combinatorial Optimization. In J.D. Schaffer, editor, *Proceedings of the Third International Conference on Genetic Algorithms*, pages 416–422. Morgan Kaufmann, San Mateo, CA, 1989.
- [176] H. Mühlenbein. Evolution in Time and Space - The Parallel Genetic Algorithm. In G.J.E. Rawlins, editor, *Foundations of Genetic Algorithm & Classifier Systems*, pages 316–338. Morgan Kaufman, San Mateo, CA, 1991.
- [177] H. Mühlenbein. Asynchronous Parallel Search by the Parallel Genetic Algorithm. In V. Ramachandran, editor, *Proceedings of the Third IEEE Symposium on Parallel and Distributed Processing*, pages 526–533. IEEE Computer Society Press, Los Alamitos, CA, 1991c.

- [178] H. Mühlenbein. Parallel Genetic Algorithms in Combinatorial Optimization. In O. Balci, R. Sharda, and S. Zenios, editors, *Computer Science and Operations Research*, pages 441–456. Pergamon Press, New York, NY, 1992.
- [179] H. Mühlenbein. How Genetic Algorithms Really Work: Mutation and Hill-Climbing. In R. Männer and B. Manderick, editors, *Parallel Problem Solving from Nature II*, pages 15–26. North-Holland, Amsterdam, 1992a.
- [180] H. Mühlenbein, M. Gorges-Schleuter, and O. Krämer. New Solutions to the Mapping Problem of Parallel Systems - the Evolution Approach. *Parallel Computing*, 6:269–279, 1987.
- [181] H. Mühlenbein, M. Gorges-Schleuter, and O. Krämer. Evolution Algorithms in Combinatorial Optimization. *Parallel Computing*, 7(1):65–85, 1988.
- [182] H. Mühlenbein, M. Schomisch, and J. Born. The Parallel Genetic Algorithm as Function Optimizer. *Parallel Computing*, 17(6-7):619–632, 1991a.
- [183] H. Mühlenbein, M. Schomisch, and J. Born. The Parallel Genetic Algorithm as Function Optimizer. In R.K. Belew and L.B. Booker, editors, *Proceedings of the Fourth International Conference on Genetic Algorithms*. Morgan Kaufmann, San Mateo, CA, 1991b.
- [184] M. Munetomo, Y. Takai, and Y. Sato. An Efficient Migration Scheme for Subpopulation-Based Asynchronous Parallel Genetic Algorithms. In S. Forrest, editor, *Proceedings of the Fifth International Conference on Genetic Algorithms*, pages 649–658. Morgan Kaufmann, San Mateo, CA, 1993.
- [185] T. Muntean and E-G. Talbi. A Parallel Genetic Algorithm for Process-Processor Mapping. In M. Durand and F. El-Dabagh, editors, *2nd Symposium on High Performance Computing*, pages 71–82. North-Holland, Amsterdam, 1991.
- [186] T.M. Nabhan and A.Y. Zomaya. A Parallel Simulated Annealing Algorithm with Low Communication Overhead. *IEEE Transactions on Parallel and Distributed Systems*, 6(12):1226–1233, 1995.
- [187] K.S. Natarajan and S. Kirkpatrick. Evaluation of Parallel Placement by Simulated Annealing: Part I - the Decomposition Approach. Report Research Report RC15246, IBM T.J. Watson Research Center, 1989.
- [188] K.S. Natarajan and S. Kirkpatrick. Evaluation of Parallel Placement by Simulated Annealing: Part II - the FLAT Approach. Report Research Report RC15247, IBM T.J. Watson Research Center, 1989.
- [189] I.H. Osman and J.P. Kelly, editors. *Meta-Heuristics: Theory & Applications*. Kluwer, Norwell, MA, 1996.
- [190] P.M. Pardalos, L. Pitsoulis, T. Mavridou, and M.G.C. Resende. Parallel Search for Combinatorial Optimization: Genetic Algorithms, Simulated Annealing, Tabu Search

- and GRASP. In A. Ferreira and J. Rolim, editors, *Proceedings of Workshop on Parallel Algorithms for Irregularly Structured Problems, Lecture Notes in Computer Science 980*, pages 317–331. Springer-Verlag, Berlin, 1995.
- [191] P.M. Pardalos, L. Pitsoulis, and M.G.C. Resende. A Parallel GRASP Implementation for the Quadratic Assignment Problem. In A. Ferreira and J. Rolim, editors, *Solving Irregular Problems in Parallel: State of the Art*. 1995.
- [192] C.B. Pettey and M.R. Leuze. A Theoretical Investigation of a Parallel Genetic Algorithm. In J.D. Schaffer, editor, *Proceedings of the Third International Conference on Genetic Algorithms*, pages 398–405. Morgan Kaufmann, San Mateo, CA, 1989.
- [193] C.B. Pettey, M.R. Leuze, and J.J. Grefenstette. A Parallel Genetic Algorithm. In J.J. Grefenstette, editor, *Proceeding of the Second International Conference on Genetic Algorithms and their Applications*, pages 155–161. Lawrence Erlbaum Associates, Hillsdale, NJ, 1987.
- [194] S.C.S. Porto, J.P.F.W. Kitajima, and C.C. Ribeiro. Performance Evaluation of a Parallel Tabu Search Task Scheduling Algorithm. *Monografias em Ciência da Computação 35/96*, Pontifícia Universidade Católica de Rio de Janeiro, 1996.
- [195] S.C.S. Porto and C.C. Ribeiro. A Tabu Search Approach to Task Scheduling on Heterogenous Processors Under Precedence Constraints. *International Journal of High-Speed Computing*, 7:207–225, 1995.
- [196] S.C.S. Porto and C.C. Ribeiro. Parallel Tabu Search Message-Passing Synchronous Strategies for Task Scheduling Under Precedence Constraints. *Journal of Heuristics*, 1(2):207–223, 1996.
- [197] D.J. Ram, T.H. Sreenivas, and K.G. Subramaniam. Parallel Simulated Annealing Algorithms. *Journal of Parallel and Distributed Computing*, 37:207–212, 1996.
- [198] C. Rego and C. Roucairol. A Parallel Tabu Search Algorithm Using Ejection Chains for the VRP. In I.H. Osman and J.P. Kelly, editors, *Meta-Heuristics: Theory & Applications*, pages 253–295. Kluwer, Norwell, MA, 1996.
- [199] Y. Rochat and É.D. Taillard. Probabilistic Diversification and Intensification in Local Search for Vehicle Routing. *Journal of Heuristics*, 1(1):147–167, 1995.
- [200] J.S. Rose, D.R. Blythe, W.M. Snelgrove, and Z.G. Vranesic. Fast, High Quality VLSI Placement on an MIMD Multiprocessor. In *Proceedings of the IEEE International Conference on Computer-Aided Design: ICCAD-86*, pages 42–45. IEEE Computer Society Press, Washington, DC, 1986.
- [201] J.S. Rose, W.M. Snelgrove, and Z.G. Vranesic. Parallel Standard Cell Placement Algorithms with Quality Equivalent to Simulated Annealing. *IEEE Transactions on Computer-Aided Design*, 9:253–259, 1990.

- [202] P. Roussel-Ragot and G. Dreyfus. A Problem-Independent Parallel Implementation of Simulated Annealing: Models and Experiments. *IEEE Transactions on Computer-Aided Design*, 9(8):827–835, 1990.
- [203] P. Roussel-Ragot and G. Dreyfus. Parallel Annealing by Multiple Trials: An Experimental Study on a Transputer Network. In R. Azencott, editor, *Simulated Annealing Parallelization Techniques*, pages 91–108. John Wiley & Sons, New York, NY, 1992.
- [204] P. Roussel-Ragot, N. Kouicem, and G. Dreyfus. Error-Free Parallel Implementation of Simulated Annealing. In H.-P. Schwefel and R. Männer, editors, *Proceedings of the 1st Workshop on Parallel Problem Solving from Nature*, pages 231–241. Springer-Verlag, Berlin, 1990.
- [205] R. Rutenbar and S. Kravitz. Layout by Annealing in a Parallel Environment. In *Proceedings of the IEEE International Conference on Computer-Aided Design: ICCAD-86*, pages 434–437. IEEE Computer Society Press, Washington, DC, 1986.
- [206] A. Sannier and E. Goodman. Genetic Learning Procedures in Distributed Environments. In J.J. Grefenstette, editor, *Proceeding of the Second International Conference on Genetic Algorithms and their Applications*, pages 162–169. Lawrence Erlbaum Associates, Hillsdale, NJ, 1987.
- [207] J.S. Sargent. *A Parallel Row-Based Algorithm with Error Control for Standard-Cell Placement on a Hypercube Multiprocessor*. PhD thesis, University of Illinois, Urbana-Illinois, 1988.
- [208] D. Schlierkamp-Voosen and H. Mühlenbein. Strategy Adaptation by Competing Subpopulations. In Y. Davidor, H.-P. Schwefel, and R. Männer, editors, *Parallel Problem Solving from Nature III, Lecture Notes in Computer Science 866*, pages 199–208. Springer-Verlag, Berlin, 1994.
- [209] V. Schnecke and O. Vornberger. An Adaptive Parallel Genetic Algorithm for VLSI-Layout Optimization. In Y. Davidor, H.-P. Schwefel, and R. Männer, editors, *Parallel Problem Solving from Nature III, Lecture Notes in Computer Science 866*, pages 859–868. Springer-Verlag, Berlin, 1996.
- [210] H.-P. Schwefel and R. Männer, editors. *Parallel Problem Solving from Nature, Lecture Notes in Computer Science 496*. Springer-Verlag, Berlin, 1991.
- [211] M. Schwehm. Implementation of Genetic Algorithms on Various Interconnection Networks. In M. Valero, E. Onate, M. Jane, J.L Larriba, and B. Suarez, editors, *Parallel Computing and Transputers Applications*, pages 195–203. Amsterdam: IOS Press, 1992.
- [212] F. Seredynski. Dynamic Mapping and Load Balancing with Parallel Genetic Algorithms. In *Proc. Int. Conf. on Machine Learning*, pages 834–839, 1994.
- [213] B. Shapiro and J. Navetta. A Massively Parallel Genetic Algorithm for RNA Secondary Structure Prediction. *The Journal of Supercomputing*, 8(3):195–207, 1994.

- [214] R. Shonkwiler. Parallel Genetic Algorithms. In S. Forrest, editor, *Proceedings of the Fifth International Conference on Genetic Algorithms*, pages 199–205. Morgan Kaufmann, San Mateo, CA, 1993.
- [215] D.J. Sirag and P.T. Weisser. Toward a Unified Thermodynamic Genetic Operator. In J.J. Grefenstette, editor, *Proceeding of the Second International Conference on Genetic Algorithms and their Applications*, pages 116–122. Lawrence Erlbaum Associates, Hillsdale, NJ, 1987.
- [216] P. Spiessens and B. Manderick. A Genetic Algorithm for Massively Parallel Computers. In R. Eckmiller, G. Hartmann, and G. Hauske, editors, *Parallel Processing in Neural Systems and Computers*, pages 31–36. North-Holland, Amsterdam, 1990.
- [217] P. Spiessens and B. Manderick. A Massively Parallel Genetic Algorithm: Implementation and First Analysis. In R.K. Belew and L.B. Booker, editors, *Proceedings of the Fourth International Conference on Genetic Algorithms*, pages 279–285. Morgan Kaufmann, San Mateo, CA, 1991.
- [218] T.J. Stanley and T. Mudge. A Parallel Genetic Algorithm for Multi-objective Microprocessor Design. In L.J. Eshelman, editor, *Proceedings of the Sixth International Conference on Genetic Algorithms*, pages 597–604. Morgan Kaufmann, San Mateo, CA, 1995.
- [219] T. Starkweather, D. Whitley, and K. Mathias. Optimization Using Distributed Genetic Algorithms. In H.-P. Schwefel and R. Männer, editors, *Parallel Problem Solving from Nature, Lecture Notes in Computer Science 496*, pages 176–185. Springer-Verlag, Berlin, 1991.
- [220] É.D. Taillard. Robust Taboo Search for the Quadratic Assignment Problem. *Parallel Computing*, 17:443–455, 1991.
- [221] É.D. Taillard. Parallel Iterative Search Methods for Vehicle Routing Problems. *Networks*, 23:661–673, 1993.
- [222] É.D. Taillard. *Recherches itératives dirigées parallèles*. PhD thesis, École Polytechnique Fédérale de Lausanne, 1993a.
- [223] É.D. Taillard. Parallel Taboo Search Techniques for the Job Shop Scheduling Problem. *ORSA Journal on Computing*, 6(2):108–117, 1994.
- [224] É.D. Taillard, P. Badeau, M. Gendreau, F. Guertin, and J.-Y. Potvin. A Tabu Search Heuristic for the Vehicle Routing Problem with Soft Time Windows. *Transportation Science*, 31:170–186, 1997.
- [225] E-G. Talbi and P. Bessière. Un algorithme génétique parallèle pour le problème de partitionnement de graphes. Technical Report RR-845-I, Institut IMAG, 1991.
- [226] E-G. Talbi and P. Bessière. A Parallel Genetic Algorithm for the Graph Partitioning Problem. In *Proceedings of the ACM International Conference on Supercomputing ICS91*, pages 312–320, 1991a.

- [227] H. Tamaki and Y. Nishikawa. A Parallel Genetic Algorithm Based on a Neighborhood Model and Its Application to Jobshop Scheduling. In R. Männer and B. Manderick, editors, *Parallel Problem Solving from Nature II*, pages 573–582. North-Holland, Amsterdam, 1992.
- [228] R. Tanase. Parallel Genetic Algorithm for a Hypercube. In J.J. Grefenstette, editor, *Proceeding of the Second International Conference on Genetic Algorithms and their Applications*, pages 177–183. Lawrence Erlbaum Associates, Hillsdale, NJ, 1987.
- [229] R. Tanase. Distributed Genetic Algorithms. In J.D. Schaffer, editor, *Proceedings of the Third International Conference on Genetic Algorithms*. Morgan Kaufmann, San Mateo, CA, 1989.
- [230] R. Tanase. *Distributed Genetic Algorithms for Function Minimization*. PhD thesis, University of Michigan, 1989.
- [231] M. Toulouse, T.G. Crainic, and M. Gendreau. Communication Issues in Designing Cooperative Multi Thread Parallel Searches. In I.H. Osman and J.P. Kelly, editors, *Meta-Heuristics: Theory & Applications*, pages 501–522. Kluwer, Norwell, MA, 1996.
- [232] M. Toulouse, T.G. Crainic, and M. Gendreau. Issues in Designing Parallel and Distributed Search Algorithms for Discrete Optimization Problems. Publication CRT-96-36, Centre de recherche sur les transports, Université de Montréal, Montréal, QC, Canada, 1996a.
- [233] M. Toulouse, T.G. Crainic, and B. Sansó. An Experimental Study of Systemic Behavior of Cooperative Search Algorithms. Publication CRT-97-26, Centre de recherche sur les transports, Université de Montréal, Montréal, QC, Canada, 1997a.
- [234] M. Toulouse, T.G. Crainic, and B. Sansó. Systemic Behavior of Cooperative Search Algorithms. Publication CRT-97, Centre de recherche sur les transports, Université de Montréal, Montréal, QC, Canada, 1997b.
- [235] H.W.J.M. Trienekens and A. de Bruin. Towards a Taxonomy of Parallel Branch and Bound Algorithms. Report EUR-CS-92-01, Department of Computer Science, Erasmus University Rotterdam, 1992.
- [236] J.M. Varanelli and J.P. Cohoon. Population-Oriented Simulated Annealing: A Genetic/Thermodynamic Hybrid Approach to Optimization. In L.J. Eshelman, editor, *Proceedings of the Sixth International Conference on Genetic Algorithms*, pages 174–181. Morgan Kaufmann, San Mateo, CA, 1995.
- [237] M.G.A. Verhoeven and E.H.L. Aarts. Parallel Local Search. *Journal of Heuristics*, 1(1):43–65, 1995.
- [238] B. Virot. Parallel Annealing by Multiple Trials: Experimental Study of a Chip Placement Problem Using a Sequent Machine. In R. Azencott, editor, *Simulated Annealing Parallelization Techniques*, pages 109–127. John Wiley & Sons, New York, NY, 1992.

- [239] H.-M. Voigt and J. Born. A Structured Distributed Genetic Algorithm for Function Optimization. In M. Grauer and D.B. Pressmar, editors, *Parallel Computing and Mathematical Optimization*, pages 199–208. Springer-Verlag, Berlin, 1990.
- [240] H.-M. Voigt, I. Santibáñez Koref, and J. Born. Hierarchical Structured Distributed Genetic Algorithms. In R. Männer and B. Manderick, editors, *Parallel Problem Solving from Nature II*, pages 155–164. North-Holland, Amsterdam, 1992.
- [241] M.D. Vose and G.E. Liepins. Punctuated Equilibria in Genetic Search. *Complex Systems*, 5:31–44, 1991.
- [242] S. Voß. Tabu Search: Applications and Prospects. In D.-Z. Du and P.M. Pardalos, editors, *Network Optimization Problems*, pages 333–353. World Scientific Publishing Co., Singapore, 1993.
- [243] D. Whitley. Cellular Genetic Algorithms. In S. Forrest, editor, *Proceedings of the Fifth International Conference on Genetic Algorithms*, pages 658–658. Morgan Kaufmann, San Mateo, CA, 1993.
- [244] D. Whitley. Applications of Modern Heuristic Methods. In V.J. Raynard-Smith, editor, *Proceedings of the Fifth International Conference on Genetic Algorithms*, pages 55–67. Alfred Waller, 1995.
- [245] D. Whitley and T. Starkweather. Optimizing Small Neural Networks Using a Distributed Genetic Algorithm. In *Proceedings of the International Conference on Neural Networks*, pages 206–209. IEEE Press, 1990a.
- [246] D. Whitley and T. Starkweather. GENITOR II: A Distributed Genetic Algorithm. *Journal of Experimental and Theoretical Artificial Intelligence*, 2(3):189–214, 1990b.
- [247] L.D. Whitley. A Genetic Algorithm Tutorial. *Statistics and Computing*, 4:65–85, 1994.
- [248] E.E. Witte, R.D. Chamberlain, and M.A. Franklin. Parallel Simulated Annealing using Speculative Computation. In *Proceedings of the 19th International Conference on Parallel Processing*, pages 286–290, 1990.
- [249] E.E. Witte, R.D. Chamberlain, and M.A. Franklin. Parallel Simulated Annealing using Speculative Computation. *IEEE Transactions on Parallel & Distributed Systems*, 2(4):483–494, 1991.
- [250] C.-P. Wong and R.-D. Fiebrich. Simulated Annealing-Based Circuit Placement on the Connection Machine System. In *Proceedings of the IEEE International Conference on Computer Design; ICCD'87*, pages 78–82. IEEE Computer Society Press, Washington, DC, 1987.