CIRRELT

Centre interuniversitaire de recherche
sur les réseaux d'entreprise, la logistique et le transport

Interuniversity Research Centre
on Enterprise Networks, Logistics and Transportation

Parallel Meta-Heuristics

Teodor Gabriel Crainic
Michel Toulouse

May 2009

CIRRELT-2009-22

Bureaux de Montréal :
Université de Montréal
C.P. 6128, succ. Centre-ville
Montréal (Québec)
Canada H3C 3J7
Téléphone : 514 343-7575
Télécopie : 514 343-7121

Bureaux de Québec :
Université Laval
Pavillon Palasis-Prince, local 2642
Québec (Québec)
Canada G1K 7P4
Téléphone : 418 656-2073
Télécopie : 418 656-2624

www.cirrelt.ca

Parallel Meta-Heuristics

Teodor Gabriel Crainic[1,2,*], Michel Toulouse[1,3]

[1] Interuniversity Research Centre on Enterprise Networks, Logistics and Transportation (CIRRELT)

[2] Department of Management and Technology, Université du Québec à Montréal, C.P. 8888, succursale Centre-ville, Montréal, Canada H3C 3P8

[3] Department of Computer Science, Oklahoma State University, 700 North Greenwood Avenue, North Hall 328, Tulsa, OK 74106-0700, USA

Abstract. We present a state-of-the-art survey of parallel meta-heuristic strategies, developments, and results. We discuss general design and implementation principles that apply to most metaheuristic classes and instantiate these principles for neighborhood and population-based metaheuristics. We also identify a number of trends and promising research directions.

Keywords. Parallel computation, parallelization strategies, meta-heuristics, cooperative search.

Acknowledgements. The authors wish to acknowledge the contributions of their colleagues and students, in particular Professor Michel Gendreau, Université de Montréal, who collaborated over the years to their work on parallel meta-heuristics for combinatorial optimization. Partial funding for this project has been provided by the Natural Sciences and Engineering Research Council of Canada (NSERC), through its Industrial Research Chair and Discovery Grants programs by the partners of the Chair, CN, Rona, Alimentation Couche-Tard and the Ministry of Transportation of Québec, and by the Fonds de recherche sur la nature et les technologies (FQRNT) through its Team Research Grants program.

---

* Corresponding author: Teodor-Gabriel.Crainic@cirrelt.ca

Dépôt légal – Bibliothèque nationale du Québec,
Bibliothèque nationale du Canada, 2009

1 Introduction

Meta-heuristics are widely acknowledged as essential tools in addressing difficult problems in numerous and diverse fields. Meta-heuristics actually often offer the only practical approach to solving complex problems of realistic dimensions.

Even using meta-heuristics, however, the limits of what may be solved in "reasonable" computing times are still reached rapidly for many problem settings, at least much too rapidly for the growing needs of research and industry alike. Heuristics do not, in general, guaranty optimality. Moreover, the performance often depends on the particular problem setting and instance characteristics. Consequently, a major issue in meta-heuristic design and calibration is not only how to build them for maximum performance, but also how to make them *robust*, in the sense of offering a consistently high level of performance over a wide variety of problem settings and characteristics.

Parallel meta-heuristics aim to address both issues. Of course, the first goal is to solve larger problem instances in reasonable computing times. In appropriate settings, such as cooperative multi-search strategies, parallel meta-heuristics also prove to be much more robust than sequential versions in dealing with differences in problem types and characteristics. They also require less extensive, and expensive, parameter calibration efforts.

The objective of this paper is to paint a general picture of the parallel meta-heuristic field. More specifically, we aim to present a state-of-the-art survey of the main parallel meta-heuristic ideas and strategies and discuss general design and implementation principles that apply to most meta-heuristic classes, to instantiate these principles for neighborhood- and population-based meta-heuristics, and to identify a number of trends and promising research directions.

The parallel meta-heuristic field is very broad, while the space available for this paper imposes hard choices and limits the presentation. In addition to the references provided in the following sections, the reader may consult a number of surveys, taxonomies, and syntheses of parallel meta-heuristics, some addressing methods based on particular methodologies, while others address the field in more comprehensive terms. Methodology-dedicated syntheses may be found in [4, 74, 75, 76, 119] for parallel simulated annealing, [2, 16, 17, 93, 104, 132] for genetic-based evolutionary methods, [26, 34, 41, 72, 151] for tabu search; [59] for scatter search, [14, 52, 81] for ant-colony methods, and [100] for Variable Neighborhood Search (VNS). Surveys and syntheses that address more than one methodology may be found in [27, 36, 37, 38, 42, 79, 84, 113, 150].

The paper is organized as follows. Section 2 is dedicated to a general discussion of basic meta-heuristics design principles, the corresponding potential for parallel computing, and the taxonomy we use to structure the presentation. Section 3 addresses strategies focusing on accelerating computing-intensive tasks without modifying the basic algorithmic design. Methods based on the decomposition of the search space are treated in Section 4, while strategies based

on the simultaneous exploration of the search space by several independent meta-heuristics constitutes the topic of Section 5. The different cooperation principles and strategies are the subject of Section 6 and we conclude in Section 7.

2 Meta-heuristics and Parallelism

This section is dedicated to an overview of the main classes of meta-heuristics and the associated potential for parallel computing. The latter is completed by a discussion of performance indicators for parallel meta-heuristics. The section concludes with the criteria used in this paper to describe and classify parallelization strategies for meta-heuristics.

2.1 Heuristics and Meta-heuristics

Given a set of objects and the contribution associated to each, an objective function computing the value of a subset of objects out of their respective contributions, and the feasibility rules specifying how subsets may be built, combinatorial optimization problems aim to select a subset of objects satisfying these rules and such that the value of the function is the highest/lowest among all possible combinations. Many problems of interest may be represented through this framework, including design, routing, and scheduling. Combinatorial optimization problems are usually formulated as (mixed) integer optimization programs. To define notation, assume that one desires to minimize an objective function $f(x)$, linear or not, subject to $x \in \mathscr{X} \subseteq \mathbb{R}^n$. The set $\mathscr{X}$ collects constraints on the *decision variables* x and defines the *feasible domain*. Decision variables are generally non-negative and all or some may be compelled to take discrete values. One seeks a globally *optimal solution* $x^* \in \mathscr{X}$ such that $f(x^*) \leq f(x)$ for all $x \in \mathscr{X}$.

In most cases, such formulations are difficult to solve for realistically-sized problem instances, the main issue being the number of feasible solutions – subsets of objects – that grows exponentially with the number of objects in the initial set. Once various methods have been applied to re-formulate the problem and bound the region where the optimal solution is to be found, most solution methods are based on some form of exploration of the set of feasible (and sometimes, infeasible) solutions. Explicit enumeration is normally out of the question and the search for the optimal solution proceeds by implicit enumeration. Branch-and-bound (and price, and cut, and ...) methods are typical of such approaches and make up one of the strategies of choice used in the search for optimal solutions to combinatorial problems. Unfortunately, these methods fail for many instances, even when parallel implementations are used. Thus, heuristics have been, and continue to be, an essential methodology in addressing combinatorial optimization formulations, often offering the only practical alternative when dealing with problem instances of realistic dimensions and characteristics.

A *heuristic* is any procedure that aims to identify a "good" feasible solution $\tilde{x} \in \mathscr{X}$. Of course, one would like $\tilde{x}$ to be identical to x^* (if the latter is unique) or $f(\tilde{x})$ to be equal to $f(x^*)$. For most heuristics, however, one can only hope (and for some, prove) that $f(\tilde{x})$ is "close" to $f(x^*)$. At the core of many heuristics, one finds an *improving* iterative procedure that *moves* from a given solution to a solution in its *neighborhood*, which is better in terms of the objective function value or some other measure based on the solution characteristics. Thus, at each iteration, such a *local search* (*LS*) procedure identifies and evaluates solutions in the neighborhood of the current solution, selects the best one relative to given criteria, and implements the transformations required to establish the selected solution as the current one. The procedure iterates until no further improvement is possible.

---

input: $x \in \mathscr{X}$ an initial solution
  $y \leftarrow Neighbor(\mathscr{N}(x))$
  while $c(y) < c(x)$
    $x \leftarrow y$
    $y \leftarrow Neighbor(\mathscr{N}(x))$
  return x

Figure 1: Local Search Template

Figure 1 illustrates the local search template (for a minimization problem). The input value x is an usually feasible initial solution. The expression $\mathscr{N}(x)$ denotes the *neighborhood* of x, that is, the set of solutions, called *neighbors*, that can be reached from x through a predefined transformation of x called *move*. Such a transformation may be simple, e.g., complement the value of an integer-valued variable, or more complex made up of a sequence of operations, e.g., λ-opt modifications of routes in vehicle routing problems (VRPs). The value of any solution x is given by $c(x)$, which may be $f(x)$, or more or less loosely related to it, or an entirely different function. Figure 2 illustrates the procedure *Neighbor*, which identifies, evaluates, and selects solutions in the neighborhood $\mathscr{N}(x)$ of the current solution x, $m(x)$ standing for a neighbor of x obtained applying the move m. The parameter *neighbor_selection* indicates whether the procedure returns the best solution in $\mathscr{N}(x)$ with respect to $c(x)$, or the first solution better than x found while exploring $\mathscr{N}(x)$. One denotes these two cases as *best-improvement* (also called *steepest descent*, or *ascent* when maximizing) and *first-improvement*, respectively.

Local search heuristics update the current solution x only when it can be improved and never backtrack to a previous solution. They therefore stop as soon as a local optimum solution is found. This inability to continue past the first encountered local optimum is a major drawback of classical heuristic schemes. Moreover, such procedures are unable to react and adapt to particular problem instances. Re-starting and randomization strategies, as well as combinations of simple heuristics offer only partial and largely unsatisfactory answers to these issues. The class of modern heuristics known as *meta-heuristics* aims to address these challenges.

input: x; *neighbor_selection*
  $NewCurrentSolution \leftarrow x$
  if (*neighbor_selection* = first improvement) **then**
    $y \leftarrow m(x)$; $\mathcal{N}(x) \leftarrow \mathcal{N}(x) \setminus y$;
    while ($c(y) \geq c(x)$ or $\mathcal{N}(x) \neq \emptyset$) $y \leftarrow m(x)$; $\mathcal{N}(x) \leftarrow \mathcal{N}(x) \setminus y$
    $NewCurrentSolution \leftarrow y$
  if (*neighbor_selection* = best improvement) **then**
    while ($\mathcal{N}(x) \neq \emptyset$)
      $y \leftarrow m(x)$; $\mathcal{N}(x) \leftarrow \mathcal{N}(x) \setminus y$
      if ($c(y) < c(NewCurrentSolution)$) **then** $NewCurrentSolution \leftarrow y$
  return $NewCurrentSolution$

Figure 2: Procedure *Neighbor*

Meta-heuristics have been defined as master strategies (heuristics) that guide and modify other heuristics to produce solutions beyond those normally identified by heuristics such as local search [69, 71]. Compared to exact search methods, such as branch-and-bound, meta-heuristics cannot generally guarantee a systematic exploration of the entire solution space. Instead, they attempt to examine only parts thereof where, according to certain criteria, one believes good solutions may be found. Well-designed meta-heuristics avoid getting trapped in local optima or sequences of visited solutions (*cycling*) and provide reasonable assurance that the search has not overlooked promising regions.

Meta-heuristics are iterative procedures, which move at each iteration toward "good" solutions in the neighborhood of the current solution or of a suitably selected subset. Unlike local search heuristics, however, meta-heuristics may move to not-necessarily-improving solutions, which constitutes the main mechanism to avoid stopping at local optima. Additional mechanisms control the evolution of the meta-heuristic to avoid cycling, learn from previous moves and encountered solutions, and provide for a thorough search. Meta-heuristics explore a *search space* that may be the feasible domain of the problem at hand, or only loosely based on it (e.g., the search space may include unfeasible solutions or may be restricted to a subset of variables only). Many meta-heuristics have been proposed. From the point of view of parallel-strategy design, however, it is convenient to discuss them according to whether their main search mechanism is based on neighborhoods or populations.

Neighborhood-based meta-heuristics implement explicitly moves to solutions selected within given neighborhoods and generally proceed following a single trajectory in the search space. Tabu search, simulated annealing, guided local search, variable neighborhood search, greedy randomized adaptive search, and iterated local search, belong to this category of meta-heuristics, and define high-level mechanisms to guide local search explorations of the search space. The high-level mechanisms monitor the status of the search, determine when particular phases

(e.g., diversification and intensification) should start, select neighborhoods and local-search procedures, etc. Figure 3 displays the general design idea for neighborhood-based meta-heuristics, emphasizing the two major nested loops of their usual implementation: An outer loop implements the high-level meta-heuristic controlling (guiding) the global search and the selected local search procedure, while the inner loop executes the local search. Notice that the Neighbor($\mathcal{N}(x)$) procedure called within the local search implicitly adds a third-level nested loop.

Identify an initial solution
while Termination criterion not satisfied
  Monitor and update global search status and
  Update meta-heuristic guidance information
    Including the selection of LS procedures and neighborhoods
  while LS termination criterion not satisfied
    $y \leftarrow$Neighbor($\mathcal{N}(x)$)

Figure 3: The Neighborhood-based Meta-heuristic Idea

Generate an initial population P of size n
while (Termination criterion not satisfied)
  Monitor and update global search status and
  Update meta-heuristic guidance information
    Including selection of elite sub-populations and
    Computation of corresponding solution values
  Compute global information (average fitness, pheromone matrix)
  Update population
    Including local search

Figure 4: The Population-based Meta-heuristic Idea

Population-based meta-heuristics use a set of solutions to concurrently sample different regions of the solution space. The search moves to new solutions by recombining elements from different solutions in the current population. We find in this group evolutionary methods (genetic algorithms), scatter search, and path relinking. For the purposes of this paper, we also include in the group ant-based methods and other swarm-based algorithms. Figure 4 display the general algorithmic idea of population-based meta-heuristics. There are several loops in an implementation of a population-based method: the generation of the initial population, the computation of solution values, which could involve the entire population, and of the global information, the generation of new individuals, which may involve local search either as an in-

dividual improvement mechanism (e.g., genetic algorithms and scatter search) or as a trajectory between two individuals (path relinking).

2.2 Sources of Parallelism

Parallel/distributed computing means that several processes work simultaneously on several processors solving a given problem instance. Parallelism thus follows from a decomposition of the total computational load and the distribution of the resulting tasks to available processors. The decomposition may concern the algorithm, the problem-instance data, or the problem structure (e.g., mathematical or attribute-based [28, 29] decomposition). In the first case, denoted *functional parallelism*, different tasks, possibly working on the "same" data, are allocated to different processors and run in parallel, possibly exchanging information. The second is denoted *data parallelism* or *domain decomposition* and refers to the case where the problem domain, or the associated search space, is decomposed and a particular solution methodology is used to address the problem on each of the resulting components of the search space. The third case is quite recent and generates tasks by decomposing the problem along sets of attributes. The decomposition could be performed through mathematical programming techniques or heuristically. Then, some tasks work on sub-problems corresponding to particular sets of attributes (i.e., part of the original search space), while others combine sub-problem solutions into whole solutions to the original problem. According to how " small" or "large" are the tasks in terms of algorithm work or search space, the parallelization is denoted *fine-* or *coarse-grained*, respectively.

From an algorithmic point of view, the main source of parallelism for meta-heuristics is the concurrent execution of their inner loop iterations: evaluating neighbors, computing the fitness of individuals, or having ants forage concurrently. Unfortunately, this is often also the only source of readily available parallelism in meta-heuristics, most other steps being time dependent and requiring the computation of the previous steps to be completed. Even when parallelism is available, synchronization enforcing the time-dependency of the meta-heuristic steps yields significant delays, which makes parallel computation non relevant.

A significant amount of parallelism may be found, on the other hand, in the domain of the problem addressed or in the corresponding search space. Indeed, there are no data dependencies between the cost or evaluation functions of different solutions and, thus, these may be computed in parallel. Furthermore, theoretically, the parallelism in the solution or search space is as large as the space itself. There are considerable limitations to an efficient exploitation of this parallelism, however. For obvious reasons, one cannot assign a processor to each solution evaluation. The solution or search space must therefore be partitioned among processors, thus serializing the evaluation of solutions assigned to the same processor. The resulting partitions are generally still too large for explicit enumeration and, thus, an exact or heuristic search method is still required for implicitly exploring it. Partitioning then raises two issues

with respect to an overall meta-heuristic search strategy. First, the control of an overall search conducted separately on several partitions of the original space and the comprehensiveness of the solution finally reached. Second, the allocation of the computing resources for an efficient exploration avoiding, for example, searching regions with poor-quality solutions. Nonetheless, besides the inner-loop computations, this is the only other relevant source of parallelism for meta-heuristics, and is exploited in many of the strategies described in this paper.

Two main approaches are used to partition the search space: *domain decomposition* and *multi search* (the name *multiple walks* is also found in the literature). The former explicitly partitions it (see Section 4), while the latter implicitly divides it through concurrent explorations by several methods, denoted in the following "search threads". Using different search strategies contributes toward a non-overlapping exploration of the search space, but does not guarantee it and, thus, a multi-search parallelization rarely provides a proper partition of the search space. Multi-search strategies, particularly those based on cooperation principles, make up the bulk of the successful parallel meta-heuristics, however. They are the object of most recent publications in the field and are addressed in Sections 5 and 6.

2.3 Parallel Meta-heuristics Strategies

We adopt the classification of Crainic and Nourredine [36], generalizing that of Crainic, Toulouse, and Gendreau [41] (see also [26, 37, 38]; [150] and [42] present classifications that proceed of the same spirit), to describe the different parallel strategies for meta-heuristics. This classification reflects the previous discussion and is sufficiently general to encompass all meta-heuristic classes, while avoiding a level of detail incompatible with the scope and dimension limits of the paper.

The three dimensions of the classification indicate how the global problem-solving process is controlled, how information is exchanged among processes, and the variety of solution methods involved in the search for solutions, respectively. The first dimension, *Search Control Cardinality*, thus specifies whether the global search is controlled by a single process or by several processes that may collaborate or not. The two alternatives are identified as *1-control (*1C*)* and *p-control (*pC*)*, respectively.

The second dimension, relative to the type of *Search Control and Communications*, addresses the issue of information exchanges. In parallel computing, one generally refers to *synchronous* and *asynchronous* communications. In the former case, all concerned processes stop and engage in some form of communication and information exchange at moments (number of iterations, time intervals, specified algorithmic stages, etc.) exogenously determined, either hard-coded or determined by a control (master) process. In the latter case, each process is in charge of its own search, as well as of establishing communications with other processes, and the global search terminates once each individual search stops. To reflect more adequately the quantity and quality of the information exchanged and shared, as well as the additional knowl-

edge derived from these exchanges (if any), we refine these notions and define four classes: *Rigid (RS)* and *Knowledge Synchronization (KS)* and, symmetrically, *Collegial (C)* and *Knowledge Collegial (KC)*.

Because more than one solution method or variant (e.g., different parameter settings) may be involved in a parallel meta-heuristic, the third dimension indicates the *Search Differentiation*: do search threads start from the same or different solutions and do they make use of the same or different search strategies? The four cases considered are: *SPSS, Same initial Point/Population, Same search Strategy*; *SPDS, Same initial Point/Population, Different search Strategies*; *MPSS, Multiple initial Points/Populations, Same search Strategies*; *MPDS, Multiple initial Points/Populations, Different search Strategies*. Obviously, one uses "point" for neighborhood-based methods, while "population" is used for genetic-based evolutionary methods, scatter search, and ant colony methods.

Based on this classification and the sources of parallelism in meta-heuristics identified at Section 2.2, we address the parallel meta-heuristic strategies in four groups and sections: 1-control strategies exploiting the intrinsic parallelism offered by the basic, inner-loop, computations of meta-heuristics in Section 3, and strategies based on explicit domain decomposition in Section 4, while Sections 5 and 6 are dedicated to independent and cooperative multi-search strategies, respectively.

We complete this section with a few notes on measures to evaluate the performances of parallel meta-heuristics The traditional goal when designing parallel solution methods is to reduce the time required to "solve", exactly or heuristically, given problem instances or to address larger instances without increasing the computational effort. For exact solution methods that run until the optimal solution is obtained, this translates into the well-known *speedup* performance measure, computed as the ratio between the wall-clock time required to solve the problem instance in parallel with p processors and the corresponding solution time of the best-known sequential algorithm; A somewhat less restrictive measure replaces the latter with the time of the parallel algorithm run on a single processor. See [8] for a detailed discussion of this issue, including additional performance measures.

Speedup measures are more difficult to define when the optimal solution is not guaranteed or the exact method is stopped before optimality is reached. Indeed, for most parallelization strategies, the sequential and parallel versions of a heuristic yield solutions that are different in value, composition, or both. Thus, an equally important objective when parallel heuristics are contemplated is to design methods that outperform their sequential counterparts in terms of solution quality and, ideally, computational efficiency, i.e., the parallel method should not require a higher overall computation effort than the sequential method or should justify the effort by higher quality solutions. Search robustness is another characteristic increasingly expected of parallel heuristics. Robustness with respect to a problem variant is meant here in the sense of providing "equally" good solutions to a large and varied set of problem instances, without excessive calibration, neither during initial development, nor when addressing new problem

instances. See [37, 38] for a discussion of these issues.

3 Low-Level 1-Control Parallelization Strategies

Parallel strategies that exploit the potential for task decomposition within the inner-loop computations of meta-heuristics are often labeled "low level" because they modify neither the algorithmic logic, nor the search space. They aim solely to accelerate the search and generally do not modify the search behavior of the sequential meta-heuristic. Typically, the exploration is initialized from a single initial solution or population, and the search proceeds according to a single meta-heuristic strategy, only the inner-loop computations being decomposed and simultaneously performed by several processors.

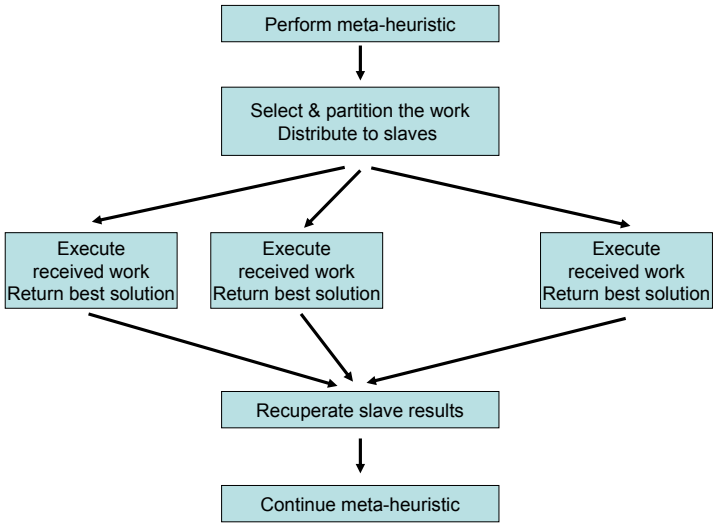

Figure 5: Low-level Decomposition Strategy

Most low-level parallel strategies belong to the 1C/RS/SPSS class and are usually implemented according to the classical master-slave parallel programming model. A "master" program executes the 1-control sequential meta-heuristic but dispatches computation-intensive tasks to be executed in parallel by "slave" programs, as illustrated in Figure 5. The master program receives and processes the information resulting from the slave operations, selects and implements moves or, for population-based methods, selects parents and generates children, updates the memories (if any) or the population, and decides whether to activate different search strategies or stop the search. The *slave* programs perform evaluations and return the results to the master which, once all the results are in, resumes the normal logic of the sequential meta-heuristic. The complete control on the algorithm execution rests with the master, which decides the work allocation for all other processors and initiates most communications.

No communications take place among slave programs. Figure 6 illustrates this control and communication scheme. Its instantiations for neighborhood and population-based methods are presented in the next two subsections.

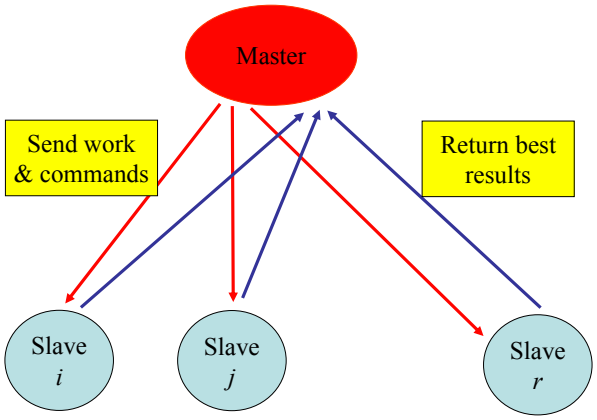

Figure 6: Master-Slave Configuration

3.1 Neighborhood-based 1C/RS/SPSS Meta-heuristics

This is the parallelization of the neighborhood evaluation procedure, depicted in Figure 2, called to compute the next current solution of a local search heuristic embedded in a neighborhood or population-based (e.g., evolutionary procedures implementing advanced "schooling" for offspring) meta-heuristic. Each iteration of the appropriate **while** loop of the procedure generates and evaluates one neighbor of the current solution, and may be executed independently of the other iterations since no data dependency exists between iterations. The computations of these iterations may then be distributed over the available p processors as illustrated in Figure 7. The master groups the neighbors into the appropriate number of tasks, which are then sent to slaves. Each slave then executes the *Neighbor* procedure on its respective part of the neighborhood and sends back the best one found. The master waits for all slaves to terminate their computations and, then, selects the best move and proceeds with the search. See [58] for an application of this strategy to a tabu search meta-heuristic for the vehicle routing problem with time-window constraints (VRPTW).

There is no predefined optimal size for the parallel tasks, the granularity $|\mathcal{N}(x)|/p$ of the decomposition depending upon the number p of available processors, as well as preoccupations with inter-processor communication times and balancing work loads among processors,

input: x a current solution; *task_size* the number of neighbors in a task
  $c(Best) \leftarrow \infty$; $number\_of\_tasks \leftarrow \frac{|\mathcal{N}(x)|}{task\_size}$; $j \leftarrow i \leftarrow 0$
  while $j < number\_of\_tasks$
    send_to_a_slave_process($task_j$); $j++$
  while $i < number\_of\_tasks$
    $y \leftarrow$ receive_from_slave(neighbor); $i++$
    if $c(y) < c(Best)$ **then** $Best \leftarrow y$
  return *Best*

Figure 7: 1C/RS Strategy: Master Neighborhood Evaluation

given the computer architecture on which computations are being performed. Thus, for example, defining each neighbor evaluation as a single task and dynamically dispatching these on a first-available, first-served basis to slave processors as they complete their tasks provides maximum flexibility and good load balancing when the evaluation of neighbors is of uneven length. For most parallel computer architectures, however, this fine-grained parallelism may come at too high an overhead cost for creating and exchanging tasks. When neighbor evaluations are sensibly the same, an often used strategy is to partition the elements defining the neighborhood into as many groups as available processors.

Notice that, when the local search procedure returns the first-best neighbor (e.g., the simple-serializable-set approach for parallel simulated annealing [82, 74]), the implemented move will often be different from that of the sequential version and, thus, the two algorithms will behave differently. Moreover, the speedup performance of this strategy will be poor when many "good" neighbors are readily available (e.g., when the temperature parameter of simulated annealing is high and most neighbors are acceptable).

3.2 Population-based 1C/RS/SPSS Meta-heuristics

1C/RS/SPSS parallelism in genetic algorithms is to be found in the loops that implement the selection, crossover, mutation, and fitness-evaluation operators, the resulting methods being variably identified in the literature as *global parallelization*, *master-slave parallelization*, and *distributed fitness evaluation*.

In theory, the degree of parallelism for each of these four operators is equal to the population size, but overhead costs may significantly decrease the degree of achievable parallelism. Actually, to be worth parallelizing, the computation must be significant. It is of little worth, for example, to parallelize a computationally simple operation like mutation. In other cases, an efficient parallelization can only be implemented on shared memory systems. Thus, for exam-

ple, the selection operator must be able to access randomly any individual in the population and its parallelization on a distributed-memory computer, where individuals are distributed across several processors, is too costly and inefficient. In practice, only the fitness evaluation can satisfy these requirements, and is often the only practical source of 1C/RS/SPSS parallelism for genetic-evolutionary methods.

The 1C/RS/SPSS parallel fitness evaluation of a population can be implemented using the master-slave model. The master partitions the individuals among slaves, which compute and return the fitness of each individual, as well as aggregate figures to facilitate the average population fitness to be computed by the master once all slaves have reported in. Similarly to other 1-control low-level parallelizations, the execution of a 1C/RS/SPSS evolutionary-genetic algorithm performs the same search as the sequential program, only faster.

The 1C/RS/SPSS parallelism for ant-colony methods lies at the level of the individual ants. Ants share information indirectly through the pheromone matrix. Furthermore, the pheromone matrix is updated once all solutions have been constructed, and there are no modifications of the pheromone matrix during a construction cycle. Consequently, the construction procedure performed by each individual ant is performed without data dependencies on the progress of the other ants.

Currently, most parallel ant-colony methods implement some form of 1C/RS/SPSS strategy according to the master-slave model, including [14, 49, 118, 120, 141]. The master builds tasks consisting of one or several ants (which can be assimilated to a "small" colony) and distributes them to the available processors. Slaves perform their construction heuristic and return their solution(s) to the master, which updates the pheromone matrix, returns it to the slaves, and so on. To further speed up computation, the pheromone update can be computed at the level of each slave, which computes the update associated to its solutions as well as the best solution and sends the aggregated pheromone update and its best solution to the master. The fine-grained version with central matrix update has been the topic of most contributions so far and, in general, it outperformed the sequential version of the algorithm. It is acknowledged, however, that it does not scale and, similarly to other meta-heuristics, this strategy is outperformed by more advanced multi-search methods.

Scatter search and path relinking implement different evolution strategies, where a restricted number of elite solutions are combined, the result being enhanced through a local search or a full-fledged meta-heuristic, usually neighborhood-based. Consequently, the 1C/RS/SPSS strategies discussed previously regarding the parallelization of local-search exploration apply straightforwardly to the present context, as in [59, 60, 62] for the p-median and the feature-selection problems.

A different 1C/RS/SPSS strategy for scatter search may be obtained by running concurrently the combination and improvement operators on several subsets of the reference set. Here, the master generates tasks by extracting a number of solution subsets and sending them

to slaves. Each slave then combines and improves its solutions, returning its results to the master for the update of the reference set. Each subset sent to a slave may contain exactly the number of solutions required by the combination operator or a higher number. In the former case [59, 60, 62], the corresponding slave performs an "iteration" of the scatter search algorithm. In the latter, several combination-improvement sequences could be executed and solutions could be returned to the master as they are found or all together at the end of all sequences. This heavy load for slaves may conduct to very different computation times and, thus, load-balancing capabilities should be added to the master.

3.3 Remarks

We complete the low-level parallelization discussion with two remarks.

A second class of low-level parallelization approaches was defined in the literature, the so-called *probing* or *look-ahead* strategies, parallelizing the *sequential fan candidate list* strategy first proposed for tabu search [72, 73]. Probing strategies belong to the 1C/KS class with any of the search-differentiation models identified previously. For neighborhood-based methods, probing may allow slaves to perform a number of iterations before synchronization and the selection of the best neighbor solution from which to proceed (one may move directly to the last solution identified by the slave or not). For population-based methods, the method may allow each slave to generate child solutions, "educate" them through a hill climbing or local-search procedure, and play out a tournament to decide who of the parents and children survive and are passed back to the master. To the best of our knowledge, [39] is the only paper ever to report results on a parallel implementation of the sequential fan candidate list strategy. The authors realized a comparative study of several synchronous tabu search parallelizations for the location-allocation problem with balancing requirements, including a straightforward 1C/RS/SPSS approach and a 1C/KS/SPSS method following the model just described (as well as a few p-control approaches). Both the 1C/KS/SPSS and the 1C/RS/SPSS heuristics yielded better solutions than sequential tabu search on the tested instances, the former being consistently superior to the latter.

We notice that a rather limited impact of low level, 1-control parallel strategies was observed in most cases. Of course, when neighborhoods are large or neighbor-evaluation procedures are costly, the corresponding gain in computing time may prove interesting, e.g., the parallel tabu searches of [19, 21, 135] for the Quadratic Assignment Problem (QAP), [20] for the Traveling Salesman Problem (TSP), and [116, 117, 115] for the task-scheduling problem. Then, when a sufficiently large number of processors is available, it might prove worthy to combine a 1C/RS/SPSS approach to more sophisticated strategies into hierarchical solution schemes (e.g., [122] were low-level parallelism accelerated the move evaluations of the individual searches engaged into an independent multi-search procedure for the VRP). More advanced multi-search strategies generally outperform low-level strategies, however.

4 Domain Decomposition

Domain or *search-space decomposition* constitutes another major parallelization strategy, one that is intuitively simple and appealing: divide the search space into smaller, usually disjoint but not necessarily exhaustive sets, solve the resulting subproblems by applying the sequential meta-heuristic on each set, collect the respective partial solutions and reconstruct an entire one.

This apparently simple idea may take several forms, however. The most straightforward approach consists in partitioning the solution vector, each resulting subset defining a subproblem. Thus, for example, the arc-design variables of a VRP may be separated into customer subsets (including the depot in each subset). A number of factors must then be specified to completely define the search-space decomposition. First, whether the partition is strict or subsets are allowed to overlap (e.g., "close by" customers may appear in two subsets in the previous VRP example). Second, whether search threads consider complete or partial solutions to the problem (in both cases, search processes access only a restricted portion of the search space). In the latter case, a complete solution has to be reconstructed at some point. Third, whether the "moves" performed on a subproblem are restricted to the corresponding search-space subset, or may involve variables in neighboring sub-spaces creating an indirect overlapping of subsets.

Strict partitioning restricts the meta-heuristic threads to their subsets and forbids moves involving solutions belonging to two or more subsets (e.g., arc swaps involving customers in different subsets). This obviously results in part of the search space being unreachable and the parallel meta-heuristic being non-optimal. Explicit or implicit overlapping aims to address this issue. But not completely and not without cost. Thus, the only way to guarantee that all potential solutions are reachable is to make overlapping cover the entire search space. This corresponds to "no decomposition" in the case of explicit overlapping and is thus not relevant. For implicit overlapping, it may also deny any gain resulting from decomposition in the first place or, in the best case, require significant overhead costs to keep most of the subproblem threads within their own subspaces.

Consequently, strict partitioning or very limited overlapping are the preferred approaches and a re-decomposition feature is included to increase the thoroughness of the search and allow all potential solutions to be examined: the decomposition is modified at regular intervals and the search is restarted using this new decomposition. This feature provides also the opportunity to define non-exhaustive decompositions, i.e., where the union of the subsets is smaller than the complete search space. A complete-solution reconstruction feature is almost always part of the procedure.

This strategy is naturally implemented using 1C/KS schemes, with a MPSS or MPDS search-differentiation strategy, according to the master-slave programming model illustrated in Figure 8. The master process determines the partition and sends subsets to slaves, synchronizes their work and collects their solutions, reconstructs solutions (if required), modifies the partitions, and determines stopping conditions. Slaves concurrently and independently perform

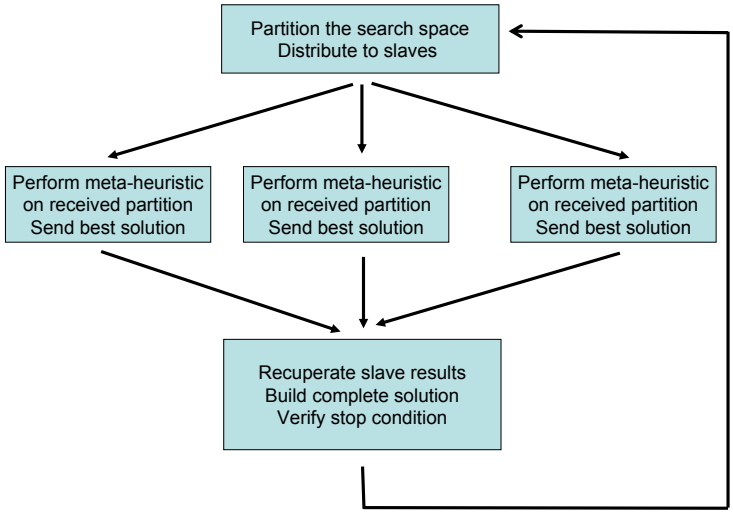

Figure 8: Domain Decomposition – Master-Slave Logic

the search on their assigned search-space subsets. The master is illustrated in Figure 9, where it is assumed that each slave performs a local search or a meta-heuristic on its subproblem and returns its current best solution when the master synchronizes activities.

---

input: $x \in \mathscr{X}$ an initial solution; $j \leftarrow i \leftarrow 0$
  Decompose problem instance into p subproblems $S_1, S_2, \ldots, S_p$
  while Stopping criteria not reached
    while $j < p$ send_to_a_slave_process(S_j, x); $j++$
    Synchronize slaves
    while $i < p$ $x_i \leftarrow$ receive_from_slave(); $i++$
    $x \leftarrow$ construct($x_1, x_2, \ldots, x_p$)
    $S_1, S_2, \ldots, S_p \leftarrow$ Modify partition
  return x

Figure 9: Domain Decomposition – Master Procedure

For neighborhood-based meta-heuristics, as well as for evolutionary methods embedding such meta-heuristics or local search procedures, one may implement the search-space decomposition approach by replacing the corresponding local search with the master-slave strategy of Figure 9. A few modifications to the original meta-heuristic may be required though. Thus, for simulated annealing, the length of the local search (cooling schedule) of each slave is reduced by a factor equal to p. The main issue for tabu search is the global tabu list the master has to reconstruct out of the local memories of the slaves simultaneously with the reconstruction of a

complete solution prior to continuing the search.

A different approach is to implementing the 1C/KS scheme of Figures 8 and 9 is to execute a full meta-heuristic on each subset of the search space, periodically modifying the partition and re-starting the search. Such an approach has been used for tabu search and proved quite successful for problems for which a large number of iterations can be performed in a relatively short time and restarting the method with a new decomposition does not require an unreasonable computational effort (e.g., [54] for the TSP, [83] for image filtering, and [68] to solve efficiently in real time several variants of the same ambulance fleet management problem instance).

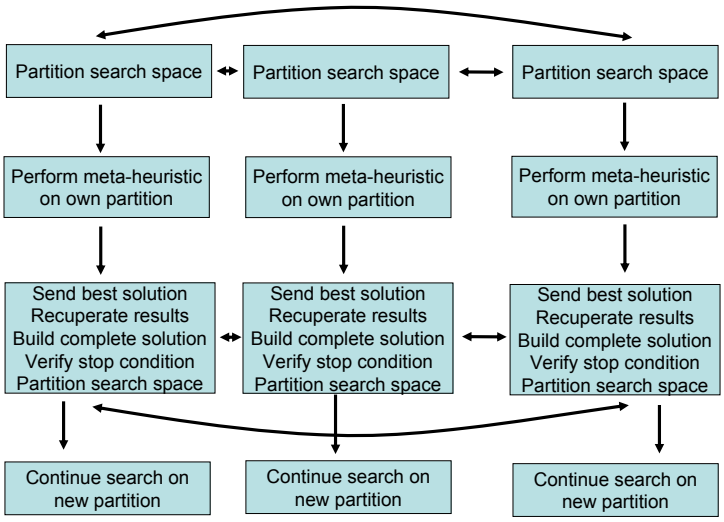

Figure 10: Domain Decomposition – Collegial Logic

We are not aware of any application of the previous approach to meta-heuristics other than tabu search, although such applications would be straightforward in most cases. A similar remark applies to the pC/KS version of search-space decomposition, a MPSS or MPDS search-differentiation strategy, illustrated in Figure 10. Such an approach was proposed in [136] for the VRP, where the customer set was partitioned, vehicles were allocated to the resulting regions, and each subproblem was solved by an independent tabu search. All processors stopped after a number of iterations that varied according to the total number of iterations already performed. The partition was then modified by an information exchange phase, during which tours, undelivered cities, and empty vehicles were exchanged between adjacent processors (corresponding to neighboring regions). At the time, this approach did allow to address successfully a number of problem instances, but the synchronization inherent in the design of the strategy hindered its performance. A parallel ant-colony approach combining this decomposition idea to a master-slave implementation was presented in [51] (parallelizing the algorithm presented in [123]), where the master generates an initial solution, defines the partition, and updates the global

pheromone matrix, while slaves execute a savings-based ant colony algorithm [124] for the resulting restricted VRP.

To complete the presentation, notice that the search behavior and the computational work performed by the sequential and parallel versions of meta-heuristics, as well as the quality of their respective solutions are not the same in most cases (again, enforcing similar behaviors would require efforts that would take away any possible benefit from parallelization). Search-space decomposition methods appear increasingly needed as the dimensions of contemplated problem instances continues to grow. Clearly, more work is required on how to best combine domain decomposition and the other parallelization strategies, cooperation in particular.

5 Independent Multi-search

Independent multi-search is among the earliest parallelization strategies. It is also the most simple and straightforward p-control parallelization strategy and generally offers very interesting performances.

The strategy consists in performing several searches simultaneously on the entire search space, starting from the same or from different initial solutions, and selecting at the end the best among the best solutions obtained by all searches. It is thus a straightforward parallelization of the well-known multi-start heuristic.

Independent multi-search methods belong to the pC/RS class of the taxonomy. No attempt is made to take advantage of the multiple search threads running in parallel other than to identify the best overall solution once all programs stop. This earns independent search strategies their rigid synchronization classification.

Independent multi-search methods turn out to be effective, simply because of the sheer quantity of computing power they allow one to apply to a given problem. This was established empirically by several papers, including the tabu searches in [10] for the QAP and [137] for the job shop scheduling problems, in which excellent results were obtained when compared to the best existing heuristics at the time. Both studies also attempted to establish some theoretical justifications for the efficiency of independent search. Battiti and Tecchiolli [10] derived models that showed that the probability of "success" increased and the corresponding average time to "success" decreased with the number of processors (provided the tabu procedure did not cycle). On the other hand, Taillard [137] showed that the conditions required for the parallel method to be "better" than the sequential one are rather strong, where "better" was defined as "the probability the parallel algorithm achieves success by time t with respect to some condition (in terms of optimality or near-optimality), is 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 different from an

exponential one, implying that independent multi-thread parallelization is an efficient strategy. The results for the job shop problem seemed to justify this claim. Similar results may also be found in [142].

This combination of simplicity of implementation and relatively good performances explains the popularity of the pC/RS/MPSS strategy for the parallelization of neighborhood-based meta-heuristics, e.g., tabu search for the VRP [122, 140] and production planning [12]; GRASP for the QAP [92, 112, 114], the Steiner problem [95, 96], and the 2-path telecommunication network design [125, 126, 127]; simulated annealing for graph partitioning [7, 6, 91, 43] (in the first two contributions the simulated annealing threads were enhanced with a simple tabu search to avoid cycling and were part of a multi-level implementation) and the TSP [99] (where each search thread was a simulated annealing procedure but an adaptive temperature schedule was controlled by a genetic algorithm); and variable neighborhood search for the p-median problem [61].

Independent multi-search pC/RS/MPSS applications to non-genetic evolutionary methods have also been proposed for scatter search [60, 62], as well as for ant-colony optimization for set covering [118], the TSP [134], and the VRP [50]. Stutzle [134] also presented a mathematical analysis and empirical results that suggest the behavior of pC/RS ant-colony parallel algorithms is similar to that of neighborhood-based meta-heuristics, i.e., providing the probability of finding the best solution is exponentially distributed with respect to time, independent multi-colony strategies are likely to find better solutions than a sequential implementation for properly selected performance targets.

In theory, pC/RS independent multi-search may be as easily adapted to parallel evolutionary meta-heuristics by running the same (MPSS) or different (MPDS) evolutionary method on disjoint populations. In practice, however, most evolutionary-genetic pC/RS parallelizations used small-sized populations, an "initial" population of size n being separated into n/p groups for the p available processors [77, 130], a strategy that did not perform well when compared to sequential algorithms with optimized population size for the particular problem instance. Indeed, while small populations speed up the computation of genetic operators such as fitness evaluation and crossover, they also display well-documented adverse impacts on the diversity of the genetic material, leading to premature convergence of the search. Running several full-sized population genetic methods in parallel [23, 24] avoids this issue and offers performances similar to those observed for the other meta-heuristics: a computation effort multiplied by the number of independent search threads and generally being outperformed by cooperative strategies.

6 Cooperative Search Strategies

Independent multi-search strategies seek to accelerate the exploration of the search space toward a better solution (compared to sequential search) by initiating simultaneous search threads from different initial points (with or without different search strategies). Cooperative search strategies go one step further and integrate mechanisms to share, *while the search is in progress*, the information obtained from this diversified exploration. The sharing and, eventually, creation of new information yields in many cases a collective output with better solutions than a parallel independent search.

Cooperative multi-search methods launch several independent *search threads*, each defining a trajectory in the search space from a possibly different initial point or population by using a possibly different meta-heuristic or search strategy. The information-sharing cooperation mechanism specifies how these independent meta-heuristics interact, the global search behavior of the cooperative parallel meta-heuristic emerging from the local interactions among them. Such similarities with systems where decisions emerge from interactions among autonomous and equal "colleagues" have inspired the name *collegial control* for the classes of strategies described in this section.

Cooperative search may be viewed as a bottom-up meta-heuristic specifying the components and their interactions, and it may thus become a "new" meta-heuristic in its own right. The key challenge of cooperation is to ensure that *meaningful* information is exchanged in a *timely* manner yielding a global parallel search that achieves a better performance than the simple concatenation of the results of the individual threads, where performance is measured in terms of computing time and solution quality. Toulouse, Crainic, and Gendreau [144] have proposed a list of fundamental issues to be addressed when designing cooperative parallel strategies for meta-heuristics: What information is exchanged? Between what processes is it exchanged? When is information exchanged? How is it exchanged? How is the imported data used? Implicit in their taxonomy and explicitly stated in later papers, the issue of whether the information is modified during exchanges or whether new information is created completes this list.

These decisions are more than implementation details, they constitute the core design parameters of a cooperative meta-heuristic. For example, a cooperative strategy could have a set of independent meta-heuristics re-start periodically from the current-best overall solution. The specification that **all** independent search threads are **re-started periodically** from the **current-best solution** of all the independent programs makes up the cooperation mechanism. It tells when programs interact (periodically; the period length is usually clearly stated), what information is exchanged (the best solutions and the overall best), between what search threads (all), what to do with the exchanged information (re-start from the imported solution).

The information to be shared among cooperating search threads should aim to improve the performance of the receiving programs, and create a global, "complete" image of the status

of the search. "Good" solutions are the most often exchanged type of information. In many cases, this takes the form of the current-best solution a search thread sends to the others or, as in the previous example, the overall best being sent to all. Not all such strategies are profitable, however.

It has been observed that sending out all current-best solutions is often counter productive, particularly when the meta-heuristic starts on a series of improving moves or generations, as solutions are generally "similar" (particularly for neighborhood-based procedures) and the receiving threads have no chance to actually act on the in-coming information. It has also been observed that always sending the overall best solution to all cooperating threads is generally bad as it rapidly decreases the diversity of the parts of the search space explored and, thus, increases the amount of worthless computational work (many threads will search in the same region) and brings an early "convergence" to a not-so-good solution. Sending out local optima only, exchanging groups of solutions, and implementing random selection procedures for the solutions to send out, the latter generally biased toward good or good-and-different solutions, are among the strategies aimed at addressing these issues. (A different strategy was proposed in [3], where the negative impact of best-solution broadcasts followed by search re-initialization was countered by having half the tabu searches regularly apply a diversification procedure, while the other half engaged in an intensification phase.)

So-called *context* information may also be exchanged. Context information refers to data collected by a meta-heuristic during its own exploration, such as the statistical information relative to the presence of particular solution elements in improving solutions (e.g., the medium and long-term memories of tabu search). Such exchanges show great promise as part of guiding heuristics for the overall search (see Section 6.3), but are not much used yet and significant research is needed to define and qualify them.

Search threads may exchange information directly or indirectly. Direct exchanges of information between two or more threads often occur when the concerned programs agree on a meeting point in time to share information. But not always. Thus, a search thread may send – broadcast – its information to one or several other threads without prior mutual agreement. Receiving search threads must then include capabilities to store such information without disturbing their own search trajectories until ready to consider it. Failure to implement such mechanisms may result in bad performances, as has been observed for strategies combining uncontrolled broadcasting of information and immediate acceptance of received data.

Indirect exchanges of information are performed through independent data structures that become "central" sources of information search threads may access asynchronously to post information and read information already posted. Such a data structure is denoted *blackboard* in computer-science and artificial-intelligence vocabulary; *memory*, *pool*, and *data warehouse* are equivalent terms found in the parallel meta-heuristic literature (due to the role assigned to the elements it contains, the terms *reference* and *elite set* are also sometimes used; in the following, we use "blackboard" for general discussions and an appropriate one among the others when

addressing specific topics). Blackboards could be centralized or distributed. Centralized blackboards have been used in most parallel meta-heuristic contributions. They post information generated by all the search threads, which, in turn, may read this information independently. The distributed approach has several blackboards located on the sites of the search threads, which thus become *hosts*, and only a subset of threads may post and access information stored on a given local blackboard. Note that a blackboard which only posts information generated by its host can support direct asynchronous interactions between the host and a subset of "adjacent" search threads. The number of blackboards in such a distributed implementation could thus be as large as the number of search threads. More complex, hierarchical structures may be contemplated, in particular for grids or loosely coupled distributed systems, but have yet to be studied.

Communications proceed according to an *interaction topology* represented by a *communication graph* specifying the processes that may engage in direct exchanges. Each node of the graph represents a search thread or a blackboard. Edges define pairs of search threads or of a search thread and a blackboard that may communicate directly. They therefore specify the direct flow of information in the cooperative system. Communication graphs may mirror the physical interconnection topology of the parallel computer executing the parallel program. Often, however, the communication graph is logically defined to suit the requirements of the cooperation strategy. Typical interaction topologies found in the parallel meta-heuristic literature are complete graphs (Figure 11), rings, grids (Figure 12), toruses, and stars (Figure 13).

When and how information is shared specifies how frequently cooperation activities are initiated and whether, in order to engage in these activities, concerned search threads must synchronize, i.e., each stopping its activities and waiting for all others to be ready, or not. One identifies these two cases as *synchronous* and *asynchronous* communications, respectively. The accumulated knowledge of the field indicates for both cases that exchanges should not be too frequent to avoid excessive communication overheads as well as premature "convergence" to local optima [145, 146, 147, 148].

Synchronous cooperation activities are initiated based on conditions external to all, or to all but one of the individual programs. In the example above, where programs interact periodically, the cooperative search strategy is synchronous, exchanges being initiated when exogenously-specified conditions, e.g., time or the number of iterations, are reached. These conditions are applied in the same way to all search threads, and communications cannot start until all have reached the designated status. The goal of synchronous cooperative strategies is to re-create a state of complete knowledge at particular points in the global search and, thus, to hopefully guide the global search into a coordinated evolution toward the desired solution to the problem. As we will see in the following subsections, this goal is rarely attained. Moreover, synchronization results in significant time inefficiencies as communications are initiated only when the slowest search thread is ready to start. We refer to such strategies as p-control, knowledge synchronous, pC/KS, with any of the SPDS, MPSS or MPDS search differentiation

approaches (appropriately applied).

A cooperation strategy is asynchronous when programs initiate cooperation activities according to their own internal state only, without coordination with other programs. Thus, for example, a search thread may make available its current best solution by posting it on a blackboard or may ask for an external solution after it failed to improve the quality on its best solution for a certain number of iterations. Asynchronous communications provide the means to build cooperation and information sharing among search threads without incuring the overheads associated to synchronization. As we will see in Sections 6.2 and 6.3, they also bring adaptivity to cooperation strategies, to the extend that the parallel cooperative meta-heuristic may react and dynamically adapt to the exploration of the search space of the particular problem instance being addressed. This is more likely to yield a globally emergent exploration strategy of the search space than synchronous approaches.

Asynchronous cooperation is fully distributed, cooperative activities being instantiated independently and concurrently by search threads, and are referred to as p-control collegial, pC/C, strategies (Section 6.2). Shared information may not just be exchanged, however, it may also be modified or used to infer knowledge. Thus, for example, statistical information may be gathered regarding configurations of solution elements in the best exchanged solutions, or the solutions gathered in a particular blackboard may form an elite population yielding new individuals to be shared among cooperating search threads. We refer to such settings as p-control knowledge collegial, pC/KC, strategies (Section 6.3).

It is worth noticing that cooperation is somewhat biased toward intensifying the search in regions of the solution space that have already been explored and where interesting solutions have been identified. This is particularly true for simple cooperation mechanisms based on synchronization or only exchanging current best solutions. It is thus important to equip the cooperation mechanisms with diversification capabilities. The introduction of probabilistic selection of exchanged solutions constitutes an example of such a mechanism. Advanced pC/KC cooperation strategies go further through creation of new solutions and guidance information as described in the following subsections.

The main principles of cooperative p-control parallelization are the same for neighborhood- and population-based meta-heuristics, even though denominations and implementation approaches may differ. One thus finds, for example, *coarse* and *fine-grained island* models for genetic-based evolutionary methods, the differentiation following from the cardinality of the population of each participating meta-heuristic, few (even down to 1 in some implementations) and many individuals, respectively. Similarly, *multi colony* is the term generally used in the ant-colony meta-heuristic community. The presentation that follows identifies these differences without dedicating subsections to each meta-heuristic class.

6.1 pC/KS Synchronous Cooperative Strategies

According to the cooperative pC/KS scheme, the independent cooperating meta-heuristics enter into an information exchange phase at pre-determined intervals, phase that must be completed before any program can restart its exploration from that synchronization point.

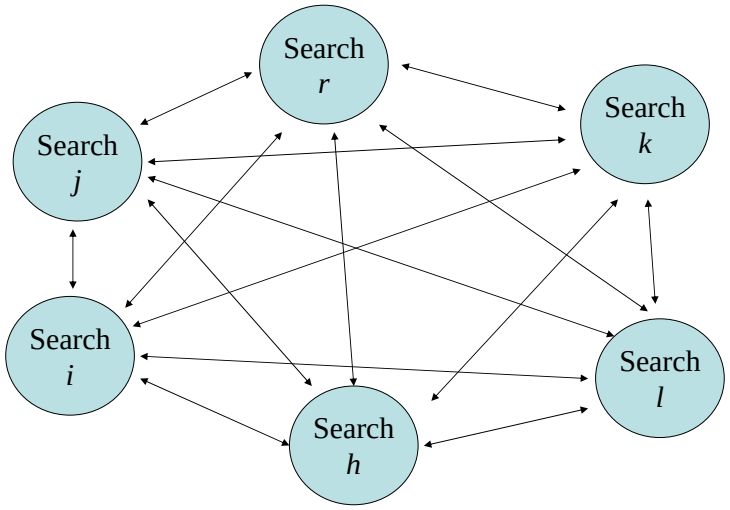

Figure 11: Complete-Graph Communication Scheme

Many proposed pC/KS cooperative search meta-heuristics followed a strategy where all threads synchronized at each point using a complete-graph communication model (Figure 11) and used master-slave implementation. In this setting, a master process, which may or not also include one of the participating meta-heuristics, initiates the other threads, stops all threads at synchronization points, gathers the sharable information, updates the global data, decides on the termination of the search and, either effectively terminates it or distributes the shared information (a good solution, generally, the overall best solution in many cases) and the continue-search signal to the other meta-heuristic threads.

The pC/KS implementation of VNS for the p-median problem proposed in [61] followed this idea, as well as the tabu search-based implementations proposed for the TSP [94], the VRP (using ejection chains) [121, 122], the QAP [46] and the task mapping problem [45], the last two contributions attempting to overcome the limitations of the master-slave setting by allowing processes, on terminating their local search phases, to synchronize and exchange best solutions with processes running on neighboring processors. A more sophisticated pC/KS approach was proposed in [108], where the master dynamically adjusted the search-strategy parameters of cooperating tabu searches according to the results each fad obtained so far. Computational results reported for the 0-1 Multi-dimensional Knapsack Problem showed that this dynamic adjustment of search parameters was indeed beneficial.

The master-slave implementation model has been applied to evolutionary methods as well.

In coarse-grained island implementations of cooperating genetic methods [43, 133], the master stops the cooperating meta-heuristics and initiates the *migration* operator to exchange among the independent populations the best or a small group of some of the best individuals in each. Applied to ant-colony systems [53] (also for a satisfiability problem), this strategy divided the colony into several sub-colonies, each assigned to a different processor. Each slave sent to the master its best solution once its ants finished searching. The master then updated the pheromone matrix and started a new search phase.

Alternatively, pC/KS cooperative schemes can be implemented by having each search thread empowered to initiate synchronization once it reaches a pre-determined status. It then broadcasts its sharable data, current best solution or group of solutions, followed by similar broadcasts performed by the other search threads. Once all information is shared, each thread performs its own import procedures on the received data and proceeds with its exploration of the search space until the next synchronization event. Such an approach was proposed for simulated annealing [48], where the search threads transmitted their best solutions every n steps, and re-started the search after updating their respective best solutions. This cooperative method outperformed an independent multi-thread search approach, both obtaining better results than the sequential version in terms of solution quality.

Most synchronous coarse-grained island parallelizations of genetic-based evolutionary methods fall under this category, where migration operators are applied at regular intervals, e.g., [152] for satisfiability problems, where the best individual of each population migrated to replace the worst of the receiving population, [55] for multi-objective telecommunication network design with migration following each generation, and [22, 23, 24, 93, 78] for graph-partitioning, the later implementing a hierarchical method, where the fitness computation was performed at the second level (through a master-slave implementation; the overhead due to the parallelization of the fitness became significant for larger numbers of processors). A similar strategy was proposed for the multi ant-colony algorithms [97, 98]. Each colony has its own pheromone matrix and may (homogeneous) or may not (heterogeneous) use the same update rule. Colonies synchronize after a fixed number of iterations to exchange elite solutions that are used to update the pheromone matrix of the receiving colony.

Several of these studies [22, 23, 24, 93] compared several implementations of coarse-grained parallel genetic methods and contributed to show that synchronous pC/KS strategies outperform independent search approaches. They also showed the superiority of dynamically-determined synchronization points, as well as that of asynchronous communications. Similar conclusions were obtained [88, 90, 89, 91] for parallel simulated annealing for the graph partitioning problem and for tabu search [39, 40] for location problems with balancing requirements.

The previous strategies are based on global exchanges of information, gathered at synchronization points during the computation and distributed to all search threads. The interest of these strategies follows from the fact that they use the best knowledge available at the synchro-

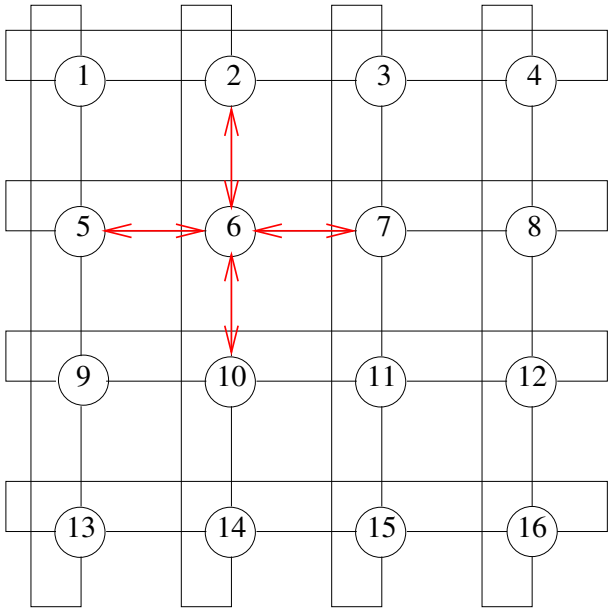

Figure 12: Grid Communication Scheme – Diffusion-based Cooperation

nization points to attempt to guide the exploration. Global information sharing has obvious drawbacks, however. When each independent program is guided by the same set of best solutions, the global search lacks diversity and, eventually, all threads will focus on the same regions of the search space. Designing globally efficient synchronous cooperative strategies has proved to be difficult in most cases.

Synchronized cooperation may alternatively be based on direct local exchanges of information, global information sharing taking place through *diffusion*. To design such a cooperative strategy, the complete communication graph used previously is replaced by a less densely connected communication topology such as ring, torus, or grid graphs. The topology restricts to a few neighbors the direct communications a search thread may engage in, as illustrated in Figure 12 for a grid communication graph where, for example, the search thread on node 6 can share information only with threads on nodes 2, 5, 7, and 10. Following synchronization, each thread continues its search based on information obtained from programs adjacent in the communication graph. Even though no direct global exchanges take place, information is still shared through diffusion. Thus, for example, assuming the meta-heuristic at node 6 received and accepted the best solution of node 2 at synchronization point i and the program at node 10 receives and accepts the solution of node 6 at synchronization point $i+1$, information from search thread on node 2 passed to the method on node 10. Through diffusion, the search threads on nodes 2 and 10 shared some information even though they are not adjacent

This idea has not been as broadly explored as the global-exchange strategy, even though synchronous cooperative mechanisms based on local exchanges and diffusion have a less negative impact on the diversity of the search-space exploration. A number of applications were

proposed for coarse-grained [15, 143] and fine-grained [56, 57, 101, 103] genetic-based evolutionary methods with good results. (It is interesting to recall that [101, 103] was part of a larger body of contributions [102, 104, 105, 106, 107] where hill-climbing heuristics were embedded into genetic algorithms to improve – "educate" – individuals and the impact of this hybridization on the behavior and performance of genetic methods was studied.) An application to ant-colony optimization methods was also proposed [98] with similar results.

Cooperation based on asynchronous information sharing generally outperform synchronous methods, however, and are the topic of the next subsection.

6.2 pC/C Asynchronous Cooperative Strategies

Historically, independent and synchronous cooperative methods were the first multi-search approaches to be developed. However, because of the shortcomings of these methods, discussed at length in the previous subsections, attention has increasingly been turned toward asynchronous strategies, which now largely define the "state-of-the-art" in parallel multi-search meta-heuristics. These asynchronous procedures all follow the same general pattern: starting from possibly different initial solutions and using possibly different search strategies, p threads explore simultaneously the search space, exchanging and, eventually, creating information according to a mechanism moved by the internal logic of each participating search thread and the state of the search. Asynchronous cooperative strategies belong either to the pC/C or to the pC/KC class of the taxonomy, the main difference between the two being whether or not any "new" knowledge is inferred on the basis of the information exchanged between the search threads; pC/KS strategies are addressed in the next subsection.

Most genetic-based evolutionary asynchronous cooperative meta-heuristics belong to the pC/C class. They generally implement a coarse-grained island model, where migration is triggered within individual populations, selected migrant individuals being directed toward either all other populations or a dynamically-selected subset. An early comparative study of coarse-grained parallel genetic methods for the graph-partitioning problem numerically showed the superiority of the pC/C strategy (with migration toward a subset of populations) over synchronous approaches [93]. Currently, this is the most popular strategy for multi-population parallel genetic methods [17].

The sharing of information in most asynchronous cooperative search strategies outside the genetic-evolutionary community is based on some form of centralized blackboard model, most often denoted *central memory* [26, 40, 41] and illustrated in Figure 13. According to this pC/C cooperation model, whenever a search thread identifies sharable information it sends it to the central memory, from where, when needed, it also retrieves information sent by the other cooperating search threads. (Information retrieval on distributed-memory systems passes through a request to the processor running the central-memory processes.) The sharable information corresponds to a locally improving solution, the most successful implementations sending new

local optima only, according to the already-mentioned information-sharing parsimony principle.

The need for cooperation is particular to each type of meta-heuristic involved in the cooperation. Following again the principle of parsimonious communications, such activities are often initiated at algorithmic steps involving a choice of solutions or a modification of the current solution from where the next local search will proceed, e.g., diversification moves in tabu search or neighborhood changes in variable neighborhood search. It may also be triggered by a priori rules, e.g., a fixed number of iterations, particularly when no such algorithmic steps exist, e.g., the tabu searches based on continuous diversification strategies. Central-memory-based asynchronous cooperative algorithms thus provide the environment of an indirect exchange of information between cooperating search threads, in particular among the one that stored the sharable information in the central memory and the one that accessed this information through a request for knowledge to the central memory.

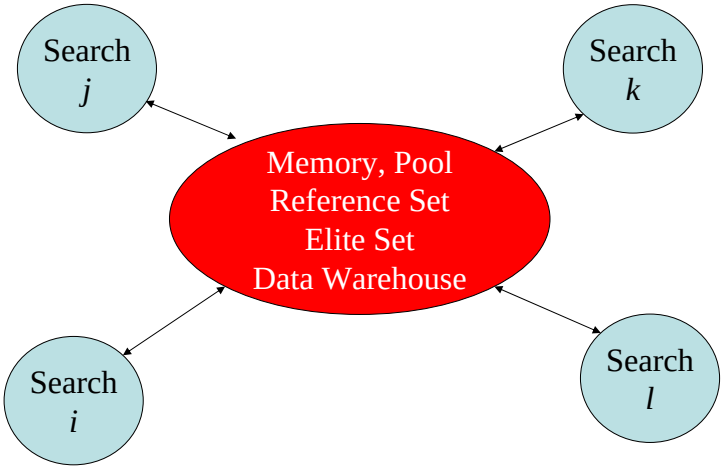

Figure 13: Star Communication Scheme – Memory-based Cooperation

Most current implementations of this approach manage the central-memory information following an algorithmic template similar to the one illustrated in Figure 14. The input variable *memory_size* specifies the capacity of the central memory, i.e., the number of solutions that can be stored in the memory. Incoming solutions are automatically accepted when the memory is not full. Acceptance is conditional to the relative interest of the incoming solution compared to the "worst" solution in the memory, otherwise. In most cases, the comparison is based on the evaluation function for the corresponding search space or the value of the objective function of the original problem. The *rWorst* solution of a full memory is then replaced with the incoming solution when the value of the latter is better (lower, for minimization problems) than that of the former. Diversity measures may modify this choice in more advanced strategies (which may also delete a larger part, half, usually, of the population in memory). The template also illustrates the random solution-extraction process, which follows the request of a cooperating search thread for a solution from memory. The random selection may be uniform or biased to

favor solutions with the best ranking based on, for example, solution values.

```
input: memory_size number of sharable information instances
  c(rWorst) ← ∞; fill ← 0
  while not finished
    x', message, pid ← receive_from_coop_search_thread()
    if message = put_in_memory then
      if c(x') < c(rWorst) then
        if fill ≤ memory_size then
          memory ← memory ∪ x'; fill ← fill + 1
        else
          memory ← memory ∪ x'
          memory ← memory \ rWorst
          rWorst ← compute_rWorst()
    if message = get_from_memory then
      rBest ← select_randomly(memory)
      send_to_search_thread(rBest, pid)
```

Figure 14: Memory Template

Several central-memory-based cooperative search strategies are described in the literature, including simulated-annealing applications to graph partitioning [88, 89, 90, 91] and the TSP [129], and the master-slave implementation of a pC/C/MPSS cooperation mechanism for VNS applied to the p-median problem [33]. In the latter method, individual VNS processes communicated exclusively with a master process., which kept, updated, and communicated the current overall best solution (it also initiated and terminated the algorithm). Solution updates and communications were performed following messages from the individual VNS threads, which proceeded with the "normal" VNS exploration for as long as the solution was improved. When the solution was not improved, it was communicated to the master (if better than the one at the last communication) and the overall best solution was requested from the master. The search was then continued starting from the best overall solution in the current neighborhood. Computational results on TSPLIB problem instances with up to 11849 customers showed that the cooperative strategy yielded significant gains in terms of computation time without loosing on solution quality, which was comparable to that of the best results in the literature (when available).

To the best of our knowledge, Crainic, Toulouse, and Gendreau were the first to propose a central-memory approach for asynchronous tabu search in their comparative study for multi-commodity location with balancing requirements [40]. Their method, where individual tabu searches sent to the memory their local-best solutions when improved and imported a probabilistically-selected (rank-biased) solution from the memory before engaging in a diversification phase, outperformed in terms of solution quality the sequential version (which was

also bested in terms of wall-clock computing time) as well as pC/RS/MPDS, pC/KS (varying the synchronization mechanisms and the Search-Differentiation strategies), and broadcast-based asynchronous pC/C cooperative strategies. The same approach was applied to the fixed cost, capacitated, multicommodity network design problem with similar results [32]. Over the last few years, several other authors have implemented fairly similar approaches to a variety of problems, including the partitioning of integrated circuits for logical testing [1], two-dimensional cutting [11], the loading of containers [13], labor-constrained scheduling [18], and VRPTW [86].

The same broad strategy was also followed when meta-heuristics belonging to different types were sequentially applied to a given problem. The two-phase approach of Gehring and Homberger for the VRPTW [63, 64, 65, 80] is a typical example of such a method, where each search thread first applies an evolution strategy to reduce the number of vehicles, followed by a tabu search to minimize the total distance traveled. A somewhat different two-phase pC/C parallel strategy was proposed in [9] for the Steiner problem, where each phase, using reactive tabu search and path relinking, respectively, implemented the pC/C asynchronous central memory strategy, all processes switching from the first to the second phase simultaneously.

The central-memory pC/C approach has proved efficient in handling the problem of premature "convergence" in cooperative search. The memory contains a large set of different solutions and cooperating search threads may import different solutions even when their cooperation activities are taking place in a short time span. Furthermore, the probabilistic solution-extraction strategies used by the central-memory program yield a cooperation that continuously evolves with respect to the search threads indirectly exchanging information.

The central-memory approach also allows for more flexibility in terms of the different meta-heuristic (and exact, eventually) methods that can be combined in a same set of cooperating programs. One can thus have methods that heuristically construct new solutions, execute neighborhood-based improving meta-heuristics, evolve populations of solutions, or perform post-optimization procedures on solutions in the memory. One can thus select cooperating methods that complement each other, as illustrated in the study of Crainic and Gendreau [31], where a genetic-method thread was added to an asynchronous multi-thread tabu search for multicommodity location-allocation with balancing requirements [40]. The tabu searches were aggressively exploring the search space, while the genetic method contributed toward increasing the diversity of solutions exchanged among the cooperating methods. The genetic method was launched once a certain number of elite solutions identified by the tabu searches were recorded in the central memory, using this memory as initial population. Asynchronous migration subsequently transfered the best solution of the genetic pool to the central memory, as well as solutions of the central memory toward the genetic population. This strategy did perform well, especially on larger instances. It also yielded an interesting observation: the best overall solution was never found by the genetic thread, but its inclusion allowed the tabu search threads to find better solutions through what appears as a more effective diversification of the global search.

Memory-based pC/C cooperative search is also computationally efficient as no costs are incurred for inter-program synchronizations. No broadcasting is taking place and there is no need for complex mechanisms to select the threads that will receive or send information and to control the cooperation. The central memory is thus an efficient implementation device that allows for a strict asynchronous mode of exchange, with no predetermined connection pattern, where no process is interrupted by another for communication purposes, but where any thread may access at all times the data previously sent out by any other search thread. These positive qualities of the central-memory cooperation concept have naturally opened the way to the development of more advanced pC/KC mechanisms where new information is generated based on the data exchanged among cooperating threads. This is the topic of the next subsection but, first, a look to a different asynchronous cooperative search strategy based on direct information exchanges among search threads.

As mentioned earlier, one can implement direct asynchronous exchanges through local memories (blackboards). Hosted by each search thread and storing information that the host makes available for sharing, such memories can be read directly and asynchronously by adjacent search threads according to their own internal logic. Information is thus shared globally through diffusion processes.

A pC/C cooperative strategy based on these ideas and denoted *multi-level cooperative search* was proposed by Toulouse, Thulasiraman, and Glover [149]. The mechanism may be instantiated with any search differentiation strategy (the authors used MPSS) and enforces the principle of controlled diffusion of information. Each search thread works at a different level of aggregation of the original problem (one processor works on the original problem) and communicates exclusively with the threads working on the immediate higher and lower aggregation levels. Improved solutions are exchanged asynchronously at various moments dynamically determined by each thread according to its own logic, status, and search history. Received solutions are used to modify the search at the receiving level. An incoming solution will not be transmitted further until a number of iterations have been performed, thus avoiding the uncontrolled diffusion of information. Excellent results have been obtained for graph and hypergraph partitioning problems [110, 111], network design [35], feature selection in biomedical data [109], and covering design [44]. It all these cases, the proposed method is either the current best or is on the par with the best meta-heuristics for the problem.

6.3 pC/KC Asynchronous Cooperative Strategies

The exchanges performed by cooperating search threads constitute a rich source of data for constructing an approximate image of the status of the global search. It has been thus widely observed that globally-optimal solutions are often similar, e.g., in the values taken by a large number of variables. Then, because the solutions exchanged are generally locally good, being local optima in many cases, one may assume that the statistical properties of the best solutions

exchanged are likely to approximate the statistical properties of the set of globally optimal solutions, and that this likelihood should increase in time with the evolution of the cooperative search. It then appears interesting to process the exchanged solutions to extract knowledge about these characteristics, and then use this knowledge to guide the search performed by the cooperating threads.

A particular form of knowledge-creation mechanism follows from the observation that the exchanged solutions form an elite population. New solutions may then be created by applying any available evolutionary meta-heuristic, solutions that improve upon their parents and thus directly enhance the global search. Moreover, these new solutions contribute to diversify the sharable information and may thus lead one or several cooperating threads to explore new regions of the search space.

Cooperative strategies including mechanisms to create new information and solutions based on the solutions exchanged belong to the p-control knowledge collegial (pC/KC) class.

Historically, two main classes of pC/KC cooperative mechanisms are found in the literature, both based on the idea of exploiting a set of elite solutions exchanged by cooperating search threads, but differing in the information that is kept in memory: *adaptive-memory* methods [128] store partial elements of good solutions and combine them to create new complete solutions that are then improved by the cooperating threads; while *central-memory* methods exchange complete elite solutions among neighborhood and population-based meta-heuristics and use them to create new solutions and knowledge to guide the cooperating threads [26, 38, 40]. The differences between the two approaches are becoming increasingly blurred, however, as the latter approach generalizes the former.

The adaptive-memory terminology was coined by Rochat and Taillard in a paper [128] proposing tabu search-based heuristics for the VRP and the VRPTW that are still among the most effective ones for both problems. (For more on adaptive-memory concepts, see [70, 138, 139]) The main idea is to keep in the memory the individual components (in routing problems, the vehicle routes) making up the elite solutions as they are found by the cooperating threads, together with memories counting the frequency of each element in the best solutions encountered so far. The elements are ranked according to the attribute values of their respective solutions, the objective value, in particular. When a cooperating thread completes its current search, it sends its best solution to the adaptive memory and, then, probabilistically selects tours in the memory to construct an initial solution for its next search. In almost all cases, the new solution will be made up of routes from different elite solutions, thus inducing a powerful diversification effect.

The adaptive-memory approach has been applied very successfully to the VRPTW [5] and [131], the latter proposing a set-covering heuristic to select the elements that will generate the new initial solution of a cooperating thread, and to real-time vehicle routing and dispatching [66], within a two-level parallelization scheme: a pC/KC/MPSS cooperating adaptive memory

scheme was implemented at the first level while, at the second level, each individual tabu search thread implemented the route decomposition of Taillard [136] with the help of several slave processors.

Badeau *et al.* [5] also reported a number of interesting findings for the development of asynchronous multi-thread procedures, whether they use adaptive memory or not. First, the performance of their method with respect to the quality of the solution was almost independent of the number of search processes (as long as this number remained within reasonable bounds) for a fixed computational effort (measured in terms of the overall number of calls to the adaptive memory by all search threads). Second, while traditional parallelization schemes rely on a one-to-one relationship between actual processors and search processes, it turned out that their method did run significantly faster when using more search processes than the number of available processors, because this allowed to overcome the bottlenecks created when several threads were trying to access simultaneously the processor on which the adaptive memory was located. Furthermore, computational evidence showed that it is not, in general, a good idea to run a search thread concurrently with the adaptive-memory-management procedure on the same processor.

Central-memory mechanisms keep full solutions, as well as attributes and context information sent by the search threads involved in cooperation. They include adaptive-memory concepts as special cases and are thus offering increased generality and flexibility. Cooperating methods may construct new solutions, execute a neighborhood-based improving meta-heuristic, implement a population-based meta-heuristic, or perform post-optimization procedures on solutions in the memory. Improving meta-heuristics aggressively explore the search space, while population-based methods (e.g., genetic algorithms [31, 86, 87] and path relinking [30]) contribute toward increasing the diversity of shared information (solutions) among the cooperating methods. Exact solution methods may participate to the cooperation either to build solutions or to seek out optimal ones (on restricted versions of the problem, eventually). Moreover, once complete solutions are stored in the central memory, statistics and information-extraction and creation mechanisms may be built based on any individual element of these solutions or combinations thereof. Memories recording the performance of individual solutions, solution components, or search threads may be added to the central memory and statistics, learning schemes, and guidance mechanisms may be gradually built.

Population-based methods, genetic algorithms, in particular, are often used to create new solutions. As described in the previous sub-section, Crainic and Gendreau [31] proposed in an early study to use the "young" set of elite solutions in the central memory to populate a genetic method and, then, to asynchronously migrate elite solutions between the two populations. The concept evolved and the set of elite solutions in the central memory is currently viewed as the population to be simultaneously evolved by the cooperating threads, including one or more evolutionary methods [30, 47, 85, 86, 87].

The cooperative meta-heuristic proposed by Le Bouthiller and Crainic [86] (also used in

[85, 87]) for the VRPTW had thus two simple genetic algorithms with order and edge recombination crossovers, respectively, evolving the same population with two tabu search methods that perform well sequentially, the Unified Tabu of Cordeau, Laporte, and Mercier [25] and Taburoute of Gendreau, Hertz, and Laporte [67]. The cooperating threads shared information about their respective good solutions identified so far. When a thread improved its best solution, it sent it to the central memory, where they were considered "in-training" until they went through the post-optimization process (2-opt, 3-opt, or-opt, and ejection-chain procedures used to reduce the number of vehicles and the total traveled distance) and become "adults". The pool of adults in the central memory formed the elite population for the genetic operators and the tabu search procedures, which required solutions when needed (at regular intervals for the Unified Tabu and at diversification time for Taburoute). This algorithm, without any calibration or tailoring, proved to be competitive with the best meta-heuristics of its day in linear speedups.

The goal of Le Bouthiller, Crainic, and Kropf [85, 87] was to improve upon this pC/C cooperative scheme by extracting new knowledge from the information exchanged, in order to guide the individual threads and, hopefully, yield a more efficient global search. The authors also aimed for a guidance mechanism independent of particular features of the problem class at hand, e.g., routes in vehicle routing problems, and thus selected to work with one of the atomic elements of the problem: the arc.

The basic idea was that an arc that appears often in good solutions and less frequently in bad solutions may be worthy of inclusion in a tentative solution, and vice versa. To implement this idea, the authors considered the frequency of inclusion of arcs in three subsets of solutions in the pool, the elite (e.g., the 10% best), average (between the 10% and 90% best), and worst (the last 10%) groups of solutions. An arc with a high frequency in a given group signals that the meta-heuristics participating to the cooperation have often produced solutions that include that arc. Patterns of arcs were then defined, representing subsets of arcs with similar frequencies of inclusion or not in particular population groups. Guidance was obtained by transmitting arc patterns to the individual threads indicating whether the arcs in the pattern should be "fixed" or "prohibited" to intensify or diversify the search, respectively ("fix" and "prohibit" were performed by using the patterns to bias the selection of arcs during moves or reproduction). The computing time allocated to the cooperative method was divided into four phases: two phases of diversification at the beginning to broaden the search, followed by two intensification phases to focus the search around promising regions. (A dynamic version of this mechanism where phases are triggered by the evolution of the population diversity and best-solution value is presented in [85]). Excellent performances in terms of solution quality and computing efficiency were observed when this pC/KC method was compared to the best-performing methods of the day.

The versatility and flexibility of the central-memory concept is also seen in the methods that start to be proposed to address so-called *rich* (combinatorial optimization) problems displaying multiple "attributes" characterizing their feasibility and optimality structures. The general approach when addressing such multi-attribute problems is to either simplify them, or to sequen-

tially solve a series of particular cases, where part of the overall problem is fixed or ignored, or both. It is well-known that this leads to suboptimal solutions, however, and thus methods are being proposed to comprehensively address rich combinatorial problems, accounting for "all" their attributes simultaneously.

According to the best knowledge of the authors, Crainic *et al.* [30] (see also [47]) were the first to propose such a methodology in the context of designing wireless networks, where seven attributes were considered simultaneously. The authors proposed a pC/KC/MPDS parallel cooperative meta-heuristic that had tabu search solvers work on limited subsets of attributes only, while a genetic method amalgamated the partial solutions attained by the tabu search procedures into complete solution to the initial problem. The global search proceeded in three phases. To initiate the search, tabu searches started from randomly-generated solutions and sent their improving solutions to the central memory to build the starting population for the evolutionary method. In the second phase, tabu search threads requested solutions at diversification time, solutions that were extracted probabilistically biased toward the best (by objective value). Finally, the third phase was activated when the global search stalled and made use of a guidance mechanism based on solution attributes different from the objective value to direct the searches of the cooperating threads toward regions of the search space where those attributes displayed desired values.

A generalization of this approach, denoted *Integrative Cooperative Search* (*ICS*) was introduced recently [28, 29]. In ICS, independent exact or meta-heuristic solution methods, the *Partial Solvers*, work on different subsets of attributes of the problem, while other algorithms, the *Integrators* combine the resulting partial solutions and improve them. Each Partial Solver is designed to investigate a subset of the problem attribute set and construct the set of corresponding elite *Partial Solutions* (more than one solver may be assigned to the same attribute subset, in which case, they are organized according to the central memory cooperation model). Each Integrator selects solutions from the different Partial-Solution sets and constructs, and possibly improves, complete solutions to the original problem, which become part of the central memory. It is noteworthy that this is a particular case of evolution, where parents and offspring are in different populations and are conceptually different. Consequently, the method is not evolving the populations of elite partial and complete solutions in the strict sense of the term, by replacing individuals. It is rather a continuous process yielding, for each Integrator thread, one or several individuals, which enrich the set of complete solutions. Partial Solvers and Integrators cooperate through the central memory and an adaptive Global Search Coordinator. The latter is a device that monitors the central memory and the information exchanged to maintain an image of the context of the global search and of the performance and evolution of each individual search thread. Guidance mechanisms are built based on this information, in particular to steer Partial Solvers toward different regions of their search subspaces.

The last two contributions belong to a development trend still in its infancy, requiring work to fully describe its behavior and characterize its performance. The preliminary results are very promising, however, and also illustrate the interest of the pC/KC central-memory asynchronous

cooperation idea.

7 Perspectives

We presented a state-of-the-art survey of the main parallel meta-heuristic ideas and strategies, discussed general design and implementation principles, and instantiated these principles for neighborhood- and population-based meta-heuristics. The survey was structured along the lines of a taxonomy of parallel meta-heuristics, which provides a rich framework for analyzing these design principles and strategies, reviewing the literature, and identifying trends and promising research directions.

To sum up, four main classes of strategies are found in the parallel meta-heuristics field: low-level decomposition of computing-intensive tasks with no modification to the original algorithm, direct decomposition of the search space, independent multi-search, and cooperative multi-search. Historically, this series corresponds to the development sequence of parallel strategies, which, initially, was proposed mainly for genetic methods, simulated annealing, and tabu search. The range of targeted meta-heuristics has broadened in recent years, multi-search strategies taking center stage. A number of studies identifying and characterizing general strategies were also proposed (see references in the Introduction) and successfully applied to various meta-heuristics and problem classes.

This is not to say that the research on parallel meta-heuristics is over. Far from it. As pointed out in the paper and summarized in the following, many open questions and challenges still face the community, both in terms of general methodology and its instantiation to the particular context of given meta-heuristics and problem classes. Indeed, such instantiations lead not only to well-adapted and performing implementations for the meta-heuristic and problem class considered, but also to a broader understanding of the methodology and its implications. This is certainly true for the more recent methods, e.g., ant-colony and other swarm-based methods, but also for the more "traditional" meta-heuristic classes where one still observes a lack of comprehensive studies focusing on these issues.

Returning to the four classes of strategies mentioned above, one should emphasize that each addresses a particular need and, therefore, they are all part of the parallel-meta-heuristic toolbox. Low-level parallelization strategies accelerate computing-intensive tasks, particularly the evaluation of a population or neighborhood, and should prove effective in addressing similar needs for swarm-inspired methods. Consequently, while the research issues are less challenging, the impact of these strategies in actual implementations may be significant, particularly as part of hierarchical strategies.

A similar case can be made for domain decomposition. Following a short period of intensive work, this strategy has been less studied in recent times. The increase in memory and

computing power of contemporary computers might explain this fact. The dimensions of the problem instances one faces keep increasing, however, and strategies that provide the means to efficiently address them are certainly needed. The decomposition of the search space of the problem at hand is a "natural" approach to attacking such problems. Research is required, however, on advanced ways to dynamically perform this decomposition and the reconstruction of whole solutions. The relations to the implicit-decomposition (attribute based) methods and the integration with cooperative-search strategies also constitute a challenging and promising research field.

Independent multi-search offers an easy access to parallel meta-heuristic computation. The straightforward parallelization of the multi-start strategy resulting in the simultaneous exploration of the search space by (possibly different) search threads starting from different initial solutions has proved its worth in numerous studies. When one looks for a "good" solution without investment in methodological development or actual coding, independent multi-search is the appropriate tool. More refined techniques, cooperative search in particular, are needed for better results, however.

Asynchronous cooperation provides a powerful, flexible and adaptable framework for parallel meta-heuristics that consistently achieved good results in terms of computing efficiency and solution quality for many meta-heuristic and problem classes. It is increasingly acknowledged as a meta-heuristic class in its own right and constitutes a rich and challenging research field. Among the many research issues of significant interest, we mention four.

First, the exchange and utilization of context data, in particular the memories local to the search threads, to construct an image of the status of the global search. Such a process may prove of great interest in better identifying the regions of the search space already visited and the ones where it would be interesting to direct the search (promising or not yet visited regions). It could also be extended to investigate the relative performance of the individual searches participating to the cooperation and either dynamically adjust the search parameters of some or even replacing an unperforming method with a better performing one.

The second issue we identify is that of learning and extracting of information from the shared data. One could, in fact, see the previous issue as a particular case of this field aimed at creating new solutions and new information, e.g., patterns of attributes in given subsets of solutions in the central memory. The goal is to 1) enrich the population of elite solutions, or parts thereof, that are shared among cooperating search threads, and 2) build guidance mechanisms that bring in a more consistent (and, sometimes, more directive) way the status of the global search to the search decisions of individual search threads. Research is this direction is still at the very beginning but has proved its worth. Of particular interest in this context are the studies aimed at the integrative cooperative methods, where the search space is indirectly partitioned according to subsets of problem attributes, and search threads address either the resulting subproblems or the integrative processes constructing and improving complete solutions.

More research is also warranted in mixing particular meta-heuristics and strategies. With respected to the latter issue, there is interest in the possible linkages between central-memory and multi-level strategies. Indeed, while it is important to preserve the multi-level data-exchange mechanisms that provides the means to control the diffusion of information, the introduction of a central memory, with guidance, eventually, could enhance the global search by, on one hand, more rapidly making available pertinent information and, on the other hand, creating new sharable data. With respect to methods, we should not only study how the parallel strategies apply to the newer meta-heuristics (e.g., swarm-based), but also how these behave while part of a cooperative algorithm and to what extent and how the behavior of the latter is modified. Of equally significant interest is the role of exact methods in cooperation.

It is noteworthy, finally, that cooperation in cooperative search methods takes place at two different levels. The first is the direct and explicit information sharing specified by the cooperation mechanism, that is, by the algorithmic design of the cooperation. In this sense, it is a top down and "local" process, exchanges taking place between particular search threads or searches and the memory, at moments determined by the algorithmic logic of search initiating communications. The second level is that of the implicit cooperation, where information spreads across all the cooperating methods through a diffusion process and correlated interactions. Implicit cooperation is bottom up and global. It is not specified algorithmically. It is rather an emergent phenomenon produced by the correlated local interactions among searches. Many research issues are related to indirect cooperation and how to harness it to enhance the optimization capabilities of cooperative meta-heuristics. The main issue is thus how to design such systems (essentially through the selection of search algorithms and direct cooperation strategies) to obtain a system-wide emergent behavior that fulfills some specific requirements, e.g., an efficient exploration of the solution space. This area of research is close to similar efforts in other fields focusing on systems that display emergent behavior through self-organization and complex adaptive behaviors, e.g., decentralized autonomic computing and social robotics. A number of concepts have been proposed in these contexts (e.g., nonlinear dynamical systems, chaos theory, attractors and system equilibrium) but have not yet resulted in significant advances for cooperative search. Research in this area will thus probably continue to be mostly empirical for some time in the future, while theoretical models are being built and put to the test. The global effort in these directions will provide us to the design tool for more powerful parallel meta-heuristics.

Acknowledgments

The authors wish to acknowledge the contributions of their colleagues and students, in particular Professor Michel Gendreau, Université de Montréal, who collaborated over the years to their work on parallel meta-heuristics for combinatorial optimization. While working on this project, the first author was the NSERC Industrial Research Chair on Logistics Management, ESG UQAM, and Adjunct Professor with the Department of Computer Science and

Operations Research, Université de Montréal, and the Department of Economics and Business Administration, Molde University College, Norway. The second author was Adjunct Professor with the Department of Computer Science and Operations Research, Université de Montréal, and Visiting Associate Professor with the Department of Computer Science at Oklahoma State University.

Partial funding for this project has been provided by the Natural Sciences and Engineering Council of Canada (NSERC), through its Industrial Research Chair and Discovery Grants programs, by the partners of the Chair, CN, Rona, Alimentation Couche-Tard and the Ministry of Transportation of Québec, and by the Fonds québécois de recherche sur la nature et les technologies (FQRNT Québec) through its Team Research grants program.

References

[1] Aiex, R.M., Martins, S.L., Ribeiro, C.C., and Rodriguez, N.R. Cooperative Multi-Thread Parallel Tabu Search with an Application to Circuit Partitioning. In *Proceedings of IRREGULAR'98 - 5th International Symposium on Solving Irregularly Structured Problems in Parallel*, volume 1457 of *Lecture Notes in Computer Science*, pages 310–331. Springer-Verlag, 1998.

[2] Alba, E., editor. *Parallel Metaheuristics: A New Class of Algorithms*. John Wiley & Sons, Hoboken, NJ, 2005.

[3] Attanasio, A., Cordeau, J.F., Ghiani, G., and Laporte, G. Parallel Tabu Search Heuristics for the Dynamic Multi-Vehicle Dial-a-Ride Problem. *Parallel Computing*, 30:377–387, 2004.

[4] Azencott, R. *Simulated Annealing Parallelization Techniques*. John Wiley & Sons, New York, NY, 1992.

[5] Badeau, P., Gendreau, M., Guertin, F., Potvin, J.-Y., and Taillard, É.D. A Parallel Tabu Search Heuristic for the Vehicle Routing Problem with Time Windows. *Transportation Research Part C: Emerging Technologies*, 5(2):109–122, 1997.

[6] Banos, R., Gil, C., Ortega, J., and Montoya, F.G. A Parallel Multilevel Metaheuristic for Graph Partitioning. *Journal of Heuristics*, 10(4):315–336, 2004.

[7] Banos, R., Gil, C., Ortega, J., and Montoya, F.G. Parallel Heuristic Search in Multilevel Graph Partitioning. In *Proceedings of the 12th Euromicro Conference on Parallel, Distributed and Network-Based Processing*, pages 88–95, 2004.

[8] Barr, R.S. and Hickman, B.L. Reporting Computational Experiments with Parallel Algorithms: Issues, Measures, and Experts Opinions. *ORSA Journal on Computing*, 5(1):2–18, 1993.

[9] Bastos, M.P. and Ribeiro, C.C. Reactive Tabu Search with Path-Relinking for the Steiner Problem in Graphs. In S. Voß, S. Martello, C. Roucairol, and Osman, I.H., editors, *Meta-Heuristics 98: Theory & Applications*, pages 31–36. Kluwer Academic Publishers, Norwell, MA, 1999.

[10] Battiti, R. and Tecchiolli, G. Parallel Based Search for Combinatorial Optimization: Genetic Algorithms and TABU. *Microprocessors and Microsystems*, 16(7):351–367, 1992.

[11] Blazewicz, J., Moret-Salvador, A., and Walkowiak, R. Parallel Tabu Search Approaches for Two-Dimentional Cutting. *Parallel Processing Letters*, 14(1):23–32, 2004.

[12] Bock, S. and Rosenberg O. A New Parallel Breadth First Tabu Search Technique for Solving Production Planning Problems. *International Transactions in Operational Research*, 7(6):625–635, 2000.

[13] Bortfeldt, A., Gehring, H., and Mack, D. A Parallel Tabu Search Algorithm for Solving the Container Loading Problem. *Parallel Computing*, 29:641–662, 2003.

[14] Bullnheimer, B., Kotsis, G., and Strauß, C. Parallelization Strategies for the Ant System. In R. De Leone, A. Murli, P. Pardalos, and G. Toraldo, editors, *High Performance Algorithms and Software in Nonlinear Optimization*, volume 24 of *Applied Optimization*, pages 87–100. Kluwer Academic Publishers, Dordrecht, 1999.

[15] Calégari, P. , Guidec, F., Kuonen, P., and Kuonen, D. Parallel Island-Based Genetic Algorithm for Radio Network Design. *Journal of Parallel and Distributed Computing*, 47(1):86–90, 1997.

[16] Cantú-Paz, E. A Survey of Parallel Genetic Algorithms. *Calculateurs Parallèles, Réseaux et Systèmes répartis*, 10(2):141–170, 1998.

[17] Cantú-Paz, E. Theory of Parallel Genetic Algorithms. In Alba, E., editor, *Parralel Metaheuristics: A New Class of Algorithms*, pages 425–445. John Wiley & Sons, Hoboken, 2005.

[18] Cavalcante, C.B.C., Cavalcante, V.F., Ribeiro, C.C., and de Souza, C.C. Parallel Cooperative Approaches for the Labor Constrained Scheduling Problem. In C.C. Ribeiro and P. Hansen, editors, *Essays and Surveys in Metaheuristics*, pages 201–225. Kluwer Academic Publishers, Norwell, MA, 2002.

[19] Chakrapani, J. and Skorin-Kapov, J. A Connectionist Approach to the Quadratic Assignment Problem. *Computers & Operations Research*, 19(3/4):287–295, 1992.

[20] Chakrapani, J. and Skorin-Kapov, J. Connection Machine Implementation of a Tabu Search Algorithm for the Traveling Salesman Problem. *Journal of Computing and Information Technology*, 1(1):29–36, 1993.

[21] Chakrapani, J. and Skorin-Kapov, J. Massively Parallel Tabu Search for the Quadratic Assignment Problem. *Annals of Operations Research*, 41:327–341, 1993.

[22] Cohoon, J., Hedge, S., Martin, W., and Richards, D. Punctuated Equilibria: A Parallel Genetic Algorithm. In J.J. Grefenstette, editor, *Proceedings of the Second International Conference on Genetic Algorithms and their Applications*, pages 148–154. Lawrence Erlbaum Associates, Hillsdale, NJ, 1987.

[23] Cohoon, J., Martin, W., and Richards, D. Genetic Algorithm and Punctuated Equilibria in VLSI. In Schwefel, H.-P. and Männer, R., editors, *Parallel Problem Solving from Nature*, volume 496 of *Lecture Notes in Computer Science*, pages 134–144. Springer-Verlag, Berlin, 1991a.

[24] Cohoon, J., Martin, W., and Richards, D. 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.

[25] Cordeau, J.-F., Laporte, G., and Mercier, A. A Unified Tabu Search Heuristic for Vehicle Routing Problems with Time Windows. *Journal of the Operational Research Society*, 52:928–936, 2001.

[26] Crainic, T.G. Parallel Computation, Co-operation, Tabu Search. In C. Rego and B. Alidaee, editors, *Metaheuristic Optimization Via Memory and Evolution: Tabu Search and Scatter Search*, pages 283–302. Kluwer Academic Publishers, Norwell, MA, 2005.

[27] Crainic, T.G. Parallel Solution Methods for Vehicle Routing Problems. In Golden, B., Raghavan, S., and Wasil, E., editors, *The Vehicle Routing Problem: Latest Advances and New Challenges*, pages 171–198. Springer, New York, 2008.

[28] Crainic, T.G., Crisan, C.C., M. Gendreau, N. Lahrichi, and W. Rei. A Concurent Evolutionary Approach for Cooperative Rich Combinatorial Optimization. In *Genetic and Evolutionary Computation Conference - GECCO 2009, July 8-12, Montréal, Canada*. ACM, 2009. forthcoming.

[29] Crainic, T.G., Crisan, C.C., M. Gendreau, N. Lahrichi, and W. Rei. Multi-thread Integrative Cooperative Optimization for Rich Combinatorial Problems. In *The 12th International Workshop on Nature Inspired Distributed Computing - NIDISC'09, 25-29 May, Rome*, 2009. forthcoming.

[30] Crainic, T.G., Di Chiara, B., Nonato, M., and Tarricone, L. Tackling Electrosmog in Completely Configured 3G Networks by Parallel Cooperative Meta-Heuristics. *IEEE Wireless Communications*, 13(6):34–41, 2006.

[31] Crainic, T.G. and Gendreau, M. Towards an Evolutionary Method - Cooperating Multi-Thread Parallel Tabu Search Hybrid. In S. Voß, S. Martello, C. Roucairol, and Osman, I.H., editors, *Meta-Heuristics 98: Theory & Applications*, pages 331–344. Kluwer Academic Publishers, Norwell, MA, 1999.

[32] Crainic, T.G. and Gendreau, M. Cooperative Parallel Tabu Search for Capacitated Network Design. *Journal of Heuristics*, 8(6):601–627, 2002.

[33] Crainic, T.G., Gendreau, M., Hansen, P., and Mladenović, N. Cooperative Parallel Variable Neighborhood Search for the *p*-Median. *Journal of Heuristics*, 10(3):293–314, 2004.

[34] Crainic, T.G., Gendreau, M., and Potvin, J.-Y. Parallel Tabu Search. In Alba, E., editor, *Parallel Metaheuristics*, pages 298–313. John Wiley & Sons, Hoboken, NJ, 2005.

[35] Crainic, T.G., Li, Y., and Toulouse, M. A First Multilevel Cooperative Algorithm for the Capacitated Multicommodity Network Design. *Computers & Operations Research*, 33(9):2602–2622, 2006.

[36] Crainic, T.G. and Nourredine, H. Parallel Meta-Heuristics Applications. In Alba, E., editor, *Parallel Metaheuristics: A New Class of Algorithms*, pages 447–494. John Wiley & Sons, Hoboken, NJ, 2005.

[37] Crainic, T.G. and Toulouse, M. Parallel Metaheuristics. In T.G. Crainic and G. Laporte, editors, *Fleet Management and Logistics*, pages 205–251. Kluwer Academic Publishers, Norwell, MA, 1998.

[38] Crainic, T.G. and Toulouse, M. Parallel Strategies for Meta-heuristics. In F. Glover and G. Kochenberger, editors, *Handbook in Metaheuristics*, pages 475–513. Kluwer Academic Publishers, Norwell, MA, 2003.

[39] Crainic, T.G., Toulouse, M., and Gendreau, M. Synchronous Tabu Search Parallelization Strategies for Multicommodity Location-Allocation with Balancing Requirements. *OR Spektrum*, 17(2/3):113–123, 1995.

[40] Crainic, T.G., Toulouse, M., and Gendreau, M. Parallel Asynchronous Tabu Search for Multicommodity Location-Allocation with Balancing Requirements. *Annals of Operations Research*, 63:277–299, 1996.

[41] Crainic, T.G., Toulouse, M., and Gendreau, M. Towards a Taxonomy of Parallel Tabu Search Algorithms. *INFORMS Journal on Computing*, 9(1):61–72, 1997.

[42] Cung, V.-D., Martins, S.L., Ribeiro, C.C., and Roucairol, C. Strategies for the Parallel Implementations of Metaheuristics. In C.C. Ribeiro and P. Hansen, editors, *Essays and Surveys in Metaheuristics*, pages 263–308. Kluwer Academic Publishers, Norwell, MA, 2002.

[43] Czech, Z.J. A Parallel Genetic Algorithm for the Set Partitioning Problem. In *8th Euromicro Workshop on Parallel and Distributed Processing*, pages 343–350, 2000.

[44] Dai, C., Li, B., and Toulouse, M. A Multilevel Cooperative Tabu Search Algorithm for the Covering Design Problem. *Journal of Combinatorial Mathematics and Combinatorial Computing*, 68:35–65, 2009.

[45] De Falco, I., Del Balio, R., and Tarantino, E. Solving the Mapping Problem by Parallel Tabu Search. Report, Istituto per la Ricerca sui Sistemi Informatici Paralleli-CNR, 1995.

[46] De Falco, I., Del Balio, R., Tarantino, E., and Vaccaro, R. Improving Search by Incorporating Evolution Principles in Parallel Tabu Search. In *Proceedings International Confonference on Machine Learning*, pages 823–828, 1994.

[47] Di Chiara, B. *Optimum Planning of 3G Cellular Systems: Radio Propagation Models and Cooperative Parallel Meta-heuristics*. PhD thesis, Dipartimento di ingegneria dell'innovatione, Universitá degli Studi di Lecce, Lecce, Italy, 2005-2006.

[48] Diekmann, R., Lüling, R., Monien, B., and Spräner, C. Combining Helpful Sets and Parallel Simulated Annealing for the Graph-Partitioning Problem. *International Journal of Parallel Programming*, 8:61–84, 1996.

[49] Doerner, K., Hartl, R.F., Kiechle, G., Lucka, M., and Reimann, M. Parallel Ant Systems for the Capacitated Vehicle Routing Problem. In Gottlieb, J. and Raidl, G.R., editors, *Evolutionary Computation in Combinatorial Optimization: 4th European Conference, EvoCOP 2004*, volume 3004 of *Lecture Notes in Computer Science*, pages 72–83. Springer-Verlag, Berlin, 2004.

[50] Doerner, K.F., Hartl, R.F., Benkner, S., and Lucka, M. Cooperative Savings Based Ant Colony Optimization - Multiple Search and Decomposition Approaches. *Parallel Processing Letters*, 16(3):351–369, 2006.

[51] Doerner, K.F., Hartl, R.F., and Lucka, M. A Parallel Version of the D-Ant Algorithm for the Vehicle Routing Problem. In Vajtersic, M., Trobec, R., Zinterhof, P., and Uhl, A., editors, *Parallel Numerics'05*, pages 109–118. Springer-Verlag, New York, NY, 2005.

[52] Dorigo, M. and Stuetzle, T. The Ant Colony Metaheuristic. Algorithms, Applications, and Advances. In F. Glover and G. Kochenberger, editors, *Handbook in Metaheuristics*, pages 251–285. Kluwer Academic Publishers, Norwell, MA, 2003.

[53] Drias, H. and Ibri , A. Parallel ACS for Weighted MAX-SAT. In Mira, J. and Álvarez, J., editors, *Artificial Neural Nets Problem Solving Methods - Proceedings of the 7th International Work-Conference on Artificial and Natural Neural Networks*, volume 2686 of *Lecture Notes in Computer Science*, pages 414–421. Springer-Verlag, Heidelberg, 2003.

[54] Fiechter, C.-N. A Parallel Tabu Search Algorithm for Large Travelling Salesman Problems. *Discrete Applied Mathematics*, 51(3):243–267, 1994.

[55] Flores, S.D., Cegla, B.B., and Caceres, D.B. Telecommunication Network Design with Parallel Multi-objective Evolutionary Algorithms. In *IFIP/ACM Latin America Networking Conference 2003*, pages –, 2003.

[56] Folino, G., Pizzuti, C., and Spezzano, G. Combining Cellular Genetic Algorithms and Local Search for Solving Satisfiability Problems. In *Proceedings of the Tenth IEEE International Conference on Tools with Artificial Intelligence*, pages 192–198. IEEE Computer Society Press, 1998.

[57] Folino, G., Pizzuti, C., and Spezzano, G. Solving the Satisfiability Problem by a Parallel Cellular Genetic Algorithm. In *Proceedings of the 24th EUROMICRO Conference*, pages 715–722. IEEE Computer Society Press, 1998.

[58] Garcia, B.L., Potvin, J.-Y., and Rousseau, J.M. A Parallel Implementation of the Tabu Search Heuristic for Vehicle Routing Problems with Time Window Constraints. *Computers & Operations Research*, 21(9):1025–1033, 1994.

[59] García-López, F., García Torres, M., Melián-Batista, B., Moreno-Pérez, J.A., and Moreno-Vega, J.M. Parallel Scatter Search. In Alba, E., editor, *Parallel Metaheuristics: A New Class of Metaheuristics*, pages 223–246. John Wiley & Sons, Hoboken, NJ, 2005.

[60] García-López, F., García Torres, M., Melián-Batista, B., Moreno-Pérez, J.A., and Moreno-Vega, J.M. Solving Feature Subset Selection Problem by a Parallel Scatter Search. *European Journal of Operational Research*, 169:477–489, 2006.

[61] García-López, F., Melián-Batista, B., Moreno-Pérez, J.A., and Moreno-Vega, J.M. The Parallel Variable Neighborhood Search for the *p*-Median Problem. *Journal of Heuristics*, 8(3):375–388, 2002.

[62] García-López, F., Melián-Batista, B., Moreno-Pérez, J.A., and Moreno-Vega, J.M. Parallelization of the Scatter Search for the *p*-Median Problem. *Parallel Computing*, 29:575–589, 2003.

[63] Gehring, H. and Homberger, J. A Parallel Hybrid Evolutionary MetaHeuristic for the Vehicle Routing Problem with Time Windows. In K. Miettinen, M.M. M akel]:a, and J. Toivanen, editors, *Proceedings of EUROGEN99 - Short Course on Evolutionary Algorithms in Engineering and Computer Science*, pages 57–64. Jyvskyla, Finland, 1997.

[64] Gehring, H. and Homberger, J. A Parallel Two-Phase Metaheuristic for Routing Problems with Time Windows. *Asia-Pacific Journal of Operational Research*, 18(1):35–47, 2001.

[65] Gehring, H. and Homberger, J. Parallelization of a Two-Phase Metaheuristic for Routing Problems with Time Windows. *Journal of Heuristics*, 8:251–276, 2002.

[66] Gendreau, M., Guertin, F., Potvin, J.-Y., and Taillard, É.D. Tabu Search for Real-Time Vehicle Routing and Dispatching. *Transportation Science*, 33(4):381–390, 1999.

[67] Gendreau, M., Hertz, A., and Laporte, G. A Tabu Search Heuristic for the Vehicle Routing Problem. *Management Science*, 40:1276–1290, 1994.

[68] Gendreau, M., Laporte , G., and Semet, F. A Dynamic Model and Parallel Tabu Search Heuristic for Real-time Ambulance Relocation. *Parallel Computing*, 27(12):1641–1653, 2001.

[69] Glover, F. Future Paths for Integer Programming and Links to Artificial Intelligence. *Computers & Operations Research*, 1(3):533–549, 1986.

[70] Glover, F. 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 Academic Publishers, Norwell, MA, 1996.

[71] Glover, F. and Laguna, M. Tabu Search. In C.R. Reeves, editor, *Modern Heuristic Techniques for Combinatorial Problems*, pages 70–150. Blackwell Scientific Publications, Oxford, 1993.

[72] Glover, F. and Laguna, M. *Tabu Search*. Kluwer Academic Publishers, Norwell, MA, 1997.

[73] Glover, F., Taillard, É.D., and de Werra, D. A User's Guide to Tabu Search. *Annals of Operations Research*, 41:3–28, 1993.

[74] Greening, D.R. A Taxonomy of Parallel Simulated Annealing Techniques. Technical Report No. RC 14884, IBM, 1989.

[75] Greening, D.R. Asynchronous Parallel Simulated Annealing. *Lectures in Complex Systems*, 3:497–505, 1990.

[76] Greening, D.R. Parallel Simulated Annealing Techniques. *Physica D*, 42:293–306, 1990.

[77] Herdy, M. Reproductive Isolation as Strategy Parameter in Hierarchical Organized Evolution Strategies. In R. Männer and B. Manderick, editors, *Parallel Problem Solving from Nature, 2*, pages 207–217. North-Holland, Amsterdam, 1992.

[78] Hidalgo, J.I., Prieto, M., Lanchares, J., Baraglia, R., Tirado, F., and Garnica, O. Hybrid Parallelization of a Compact Genetic Algorithm. In *Proceedings of the 11th uromicro Conference on Parallel, Distributed and Network-Based Processing*, pages 449–455, 2003.

[79] Holmqvist, K., Migdalas, A., and Pardalos, P.M. Parallelized Heuristics for Combinatorial Search. In A. Migdalas, P.M. Pardalos, and S. Storoy, editors, *Parallel Computing in Optimization*, pages 269–294. Kluwer Academic Publishers, Norwell, MA, 1997.

[80] Homberger, J. and Gehring, H. Two Evolutionary Metaheuristics for the Vehicle Routing Problem with Time Windows. *INFOR*, 37:297–318, 1999.

[81] Janson, S., Merkle, D., and Middendorf, M. Parallel Ant Colony Algorithms. In Alba, E., editor, *Parallel Metaheuristics: A New Class of Metaheuristics*, pages 171–201. John Wiley & Sons, Hoboken, NJ, 2005.

[82] Kravitz, S.A. and Rutenbar, R. Placement by Simulated Annealing on a Multiprocessor. *IEEE Transactions on Computer-Aided Design*, 6:534–549, 1987.

[83] Laganière, R. and Mitiche, A. Parallel Tabu Search for Robust Image Filtering. In *Proceedings of IEEE Workshop on Nonlinear Signal and Image Processing (NSIP'95)*, volume 2, pages 603–605, 1995.

[84] Laursen, P.S. Parallel Heuristic Search – Introductions and a New Approach. In A. Ferreira and P.M. Pardalos, editors, *Solving Combinatorial Optimization Problems in Parallel*, volume 1054 of *Lecture Notes in Computer Science 1054*, pages 248–274. Springer-Verlag, Berlin, 1996.

[85] Le Bouthillier, A. *Recherches coopératives pour la résolution de problèmes d'optimisation combinatoire*. PhD thesis, Département d'informatique et de recherche opérationnelle, Université de Montréal, Montréal, QC, Canada, 2007.

[86] Le Bouthillier, A. and Crainic, T.G. A Cooperative Parallel Meta-Heuristic for the Vehicle Routing Problem with Time Windows. *Computers & Operations Research*, 32(7):1685–1708, 2005.

[87] Le Bouthillier, A., Crainic, T.G., and Kropf, P. A Guided Cooperative Search for the Vehicle Routing Problem with Time Windows. *IEEE Intelligent Systems*, 20(4):36–42, 2005.

[88] Lee, K.G. and Lee, S.Y. Efficient Parallelization of Simulated Annealing Using Multiple Markov Chains: An Application to Graph Partitioning. In Mudge, T.N., editor, *Proceedings of the International Conference on Parallel Processing*, volume III: Algorithms and Applications, pages 177–180. CRC Press, 1992.

[89] Lee, K.G. and Lee, S.Y. Synchronous and Asynchronous Parallel Simulated Annealing with Multiple Markov Chains. In F.J. Brandenburg, editor, *Graph Drawing - Proceedings GD '95, Symposium on Graph Drawing, Passau, Germany*, volume 1027 of *Lecture Notes in Computer Science*, pages 396–408. Springer-Verlag, Berlin, 1995.

[90] Lee, S.Y. and Lee, K.G. Asynchronous Communication of Multiple Markov Chains in Parallel Simulated Annealing. In Mudge, T.N., editor, *Proceedings of the International Conference on Parallel Processing*, volume III: Algorithms and Applications, pages 169–176. CRC Press, Boca Raton, FL, 1992.

[91] Lee, S.Y. and Lee, K.G. Synchronous and Asynchronous Parallel Simulated Annealing with Multiple Markov Chains. *IEEE Transactions on Parallel and Distributed Systems*, 7(10):993–1007, 1996.

[92] Li, Y., Pardalos, P.M., and Resende, M.G.C. A Greedy Randomized Adaptive Search Procedure for Quadratic Assignment Problem. In *DIMACS Implementation Challenge, DIMACS Series on Discrete Mathematics and Theoretical Computer Science*, volume 16, pages 237–261. American Mathematical Society, 1994.

[93] Lin, S.-C., Punch, W., and Goodman, E. 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.

[94] Malek, M., Guruswamy, M., Pandya, M., and Owens, H. Serial and Parallel Simulated Annealing and Tabu Search Algorithms for the Traveling Salesman Problem. *Annals of Operations Research*, 21:59–84, 1989.

[95] Martins, S.L., Resende, M.G.C., Ribeiro, C.C., and Parlados, P.M. A Parallel Grasp for the Steiner Tree Problem in Graphs Using a Hybrid Local Search Strategy. *Journal of Global Optimization*, 17:267–283, 2000.

[96] Martins, S.L., Ribeiro, C.C., and Souza, M.C. A Parallel GRASP for the Steiner Problem in Graphs. In A. Ferreira and J. Rolim, editors, *Proceedings of IRREGULAR'98 – 5th International Symposium on Solving Irregularly Structured Problems in Parallel*, volume 1457 of *Lecture Notes in Computer Science*, pages 285–297. Springer-Verlag, 1998.

[97] Michels, R. and Middendorf, M. An Ant System for the Shortest Common Supersequence Problem. In D. Corne, M. Dorigo, and F. Glover, editors, *New Ideas in Optimization*, pages 51–61. McGraw-Hill, 1999.

[98] Middendorf, M., Reischle, F., and Schmeck, S. Multi Colony Ant Algorithms. *Journal of Heuristics*, 8(3):305–320, 2002.

[99] Miki, M., Hiroyasu, T., Wako, J., and Yoshida, T. Adaptive Temperature Schedule Determined by Genetic Algorithm for Parallel Simulated Annealing. In *CEC'03 - The 2003 Congress on Evolutionary Computation*, volume 1, pages 459–466, 2003.

[100] Moreno-Pérez, J.A., Hansen, P., and Mladenović, N. Parallel Variable Neighborhood Search. In Alba, E., editor, *Parallel Metaheuristics: A New Class of Metaheuristics*, pages 247–266. John Wiley & Sons, Hoboken, NJ, 2005.

[101] Mühlenbein, H. Parallel Genetic Algorithms, Population Genetics and Combinatorial Optimization. In J.D. Schaffer, editor, *Proceedings of the Third International Conference on Genetic Algorithms*, pages 416–421. Morgan Kaufmann, San Mateo, CA, 1989.

[102] Mühlenbein, H. 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.

[103] Mühlenbein, H. Parallel Genetic Algorithms, Population Genetics, and Combinatorial Optimization. In Becker, J.D., I. Eisele, and Mündemann, F.W., editors, *Parallelism, Learning, Evolution. Workshop on Evolutionary Models and Strategies - WOPPLOT 89*, pages 398–406. Springer-Verlag, Berlin, 1991.

[104] Mühlenbein, H. Parallel Genetic Algorithms in Combinatorial Optimization. In O. Balci, R. Sharda, and S. Zenios, editors, *Computer Science and Operations Research: New Developments in their Interface*, pages 441–456. Pergamon Press, New York, NY, 1992.

[105] Mühlenbein, H. How Genetic Algorithms Really Work: Mutation and Hill-Climbing. In R. Männer and B. Manderick, editors, *Parallel Problem Solving from Nature, 2*, pages 15–26. North-Holland, Amsterdam, 1992a.

[106] Mühlenbein, H., Gorges-Schleuter, M., and Krämer, O. New Solutions to the Mapping Problem of Parallel Systems - the Evolution Approach. *Parallel Computing*, 6:269–279, 1987.

[107] Mühlenbein, H., Gorges-Schleuter, M., and Krämer, O. Evolution Algorithms in Combinatorial Optimization. *Parallel Computing*, 7(1):65–85, 1988.

[108] Niar, S. and Fréville, A. A Parallel Tabu Search Algorithm For The 0-1 Multidimensional Knapsack Problem. In *11th International Parallel Processing Symposium (IPPS '97), Geneva, Switzerland*, pages 512–516. IEEE, 1997.

[109] Oduntan, I.O., Toulouse, M., Baumgartner, R., Bowman, C., Somorjai, R., and Crainic, T.G. A Multilevel Tabu Search Algorithm for the Feature Selection Problem in Biomedical Data Sets. *Computers & Mathematics with Applications*, 55(5):1019–1033, 2008.

[110] Ouyang, M., Toulouse, M., Thulasiraman, K., Glover, F., and Deogun, J.S. Multi-Level Cooperative Search: Application to the Netlist/Hypergraph Partitioning Problem. In *Proceedings of International Symposium on Physical Design*, pages 192–198. ACM Press, 2000.

[111] Ouyang, M., Toulouse, M., Thulasiraman, K., Glover, F., and Deogun, J.S. Multilevel Cooperative Search for the Circuit/Hypergraph Partitioning Problem. *IEEE Transactions on Computer-Aided Design*, 21(6):685–693, 2002.

[112] Pardalos, P.M., Li, Y., and Murthy, K.A. Computational Experience with Parallel Algorithms for Solving the Quadratic Assignment Problem. In O. Balci, R. Sharda, and S. Zenios, editors, *Computer Science and Operations Research: New Developments in their Interface*, pages 267–278. Pergamon Press, New York, NY, 1992.

[113] Pardalos, P.M., L. Pitsoulis, T. Mavridou, and Resende, M.G.C. 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*, volume 980, pages 317–331. Springer-Verlag, Berlin, 1995.

[114] Pardalos, P.M., Pitsoulis, L., and Resende, M.G.C. 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*, pages 115–130. Kluwer Academic Publishers, Norwell, MA, 1995.

[115] Porto, S.C.S., Kitajima, J.P.F.W., and Ribeiro, C.C. Performance Evaluation of a Parallel Tabu Search Task Scheduling Algorithm. *Parallel Computing*, 26:73–90, 2000.

[116] Porto, S.C.S. and Ribeiro, C.C. A Tabu Search Approach to Task Scheduling on Heteregenous Processors Under Precedence Constraints. *International Journal of High-Speed Computing*, 7:45–71, 1995.

[117] Porto, S.C.S. and Ribeiro, C.C. Parallel Tabu Search Message-Passing Synchronous Strategies for Task Scheduling Under Precedence Constraints. *Journal of Heuristics*, 1(2):207–223, 1996.

[118] Rahoual, M., Hadji, R., and Bachelet, V. Parallel Ant System for the Set Covering Problem. In M. Dorigo, G. Di Caro, and M. Sampels, editors, *Ant Algorithms - Proceedings of the Third International Workshop, ANTS 2002*, volume 2463 of *Lecture Notes in Computer Science*, pages 262–267. Springer-Verlag, Berlin, 2002.

[119] Ram, D.J., Sreenivas, T.H., and Subramaniam, K.G. Parallel Simulated Annealing Algorithms. *Journal of Parallel and Distributed Computing*, 37:207–212, 1996.

[120] Randall, M. and Lewis, A. A Parallel Implementation of Ant Colony Optimisation. *Journal of Parallel and Distributed Computing*, 62:1421–1432, 2002.

[121] Rego, C. Node Ejection Chains for the Vehicle Routing Problem: Sequential and Parallel Algorithms. *Parallel Computing*, 27:201–222, 2001.

[122] Rego, C. and Roucairol, C. 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 Academic Publishers, Norwell, MA, 1996.

[123] Reimann, M., Doerner, K., and Hartl, R. D-Ants: Savings Based Ants Divide and Conquer the Vehicle Routing Problem. *Computers & Operations Research*, 31(4):563–591, 2004.

[124] Reimann, M., Stummer, M., and Doerner, K. A Savings Based Ants System for the Vehicle Routing Problem. In Langton, C., Cantú-Paz, E., Mathias, K.E., Roy, R., Davis, L., Poli, R., Balakrishnan, K., Honavar, V., Rudolph, G., Wegener, J., Bull, L., Potter, M.A., Schultz, A.C., Miller, J.F., Burke, E.K., and Jonoska. N., editors, *GECCO 2002: Proceedings of the Genetic and Evolutionary Computation Conference, New York, USA, July 9-13, 2002*, pages 1317–1326. Morgan Kaufmann Publishers, Inc., San Francisco, CA, 2002.

[125] Ribeiro, C.C. and Rosseti, I. A Parallel GRASP Heuristic for the 2-path Network Design Problem. 4 journée ROADEF, Paris, February 20-22, 2002.

[126] Ribeiro C.C. and Rosseti, I. A Parallel GRASP Heuristic for the 2-path Network Design Problem. Third Meeting of the PAREO Euro Working Group, Guadeloupe (Frane), May, 2002.

[127] Ribeiro C.C. and Rosseti, I. Parallel grasp with path-relinking heuristic for the 2-path network design problem. AIRO'2002, L'Aquila, Italy, September, 2002.

[128] Rochat, Y. and Taillard, É.D. Probabilistic Diversification and Intensification in Local Search for Vehicle Routing. *Journal of Heuristics*, 1(1):147–167, 1995.

[129] Sanvicente-Sánchez, H. and Frausto-Solís, J. MPSA: A Methodology to Parallelize Simulated Annealing and Its Application to the Traveling Salesman Problem. In C.A. Coello Coello, A. de Albornoz, L.E. Sucar, and O.C. Battistutti, editors, *MICAI 2002: Advances in Artificial Intelligence*, volume 2313 of *Lecture Notes in Computer Science*, pages 89–97. Springer-Verlag Heidelberg, 2002.

[130] Schlierkamp-Voosen, D. and Mühlenbein, H. Strategy Adaptation by Competing Subpopulations. In Y. Davidor, Schwefel, H.-P., and Männer, R., editors, *Parallel Problem Solving from Nature III*, volume 866 of *Lecture Notes in Computer Science*, pages 199–208. Springer-Verlag, Berlin, 1994.

[131] Schulze, J. and Fahle, T. A Parallel Algorithm for the Vehicle Routing Problem with Time Window Constraints. *Annals of Operations Research*, 86:585–607, 1999.

[132] Shonkwiler, R. 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.

[133] Solar, M., Parada, V., and Urrutia, R. A Parallel Genetic Algorithm to Solve the Set-Covering Problem. *Computers & Operations Research*, 29(9):1221–1235, 2002.

[134] Stutzle, T. Parallelization Strategies for Ant Colony Optimization. In Eiben, A.E., Back, T., Schoenauer, M., and Schwefel, H.-P., editors, *Proceedings of Parallel Problem Solving from Nature V*, volume 1498 of *Lecture Notes in Computer Science*, pages 722–731. Springer-Verlag, Heidelberg, 1998.

[135] Taillard, É.D. Robust Taboo Search for the Quadratic Assignment Problem. *Parallel Computing*, 17:443–455, 1991.

[136] Taillard, É.D. Parallel Iterative Search Methods for Vehicle Routing Problems. *Networks*, 23:661–673, 1993.

[137] Taillard, É.D. Parallel Taboo Search Techniques for the Job Shop Scheduling Problem. *ORSA Journal on Computing*, 6(2):108–117, 1994.

[138] Taillard, É.D., Gambardella, L.M., Gendreau, M., and M., Potvin, J.-Y. Adaptive Memory Programming: A Unified View of Metaheuristics. *European Journal of Operational Research*, 135:1–10, 1997.

[139] Taillard, É.D., Gambardella, L.M., Gendreau, M., and M., Potvin, J.-Y. Programmation à mémoire adaptative. *Calculateurs Parallèles, Réseaux et Systèmes répartis*, 10:117–140, 1998.

[140] Talbi, E.-G., Hafidi, Z., and Geib, J.-M. Parallel Adaptive Tabu Search Approach. *Parallel Computing*, 24:2003–2019, 1998.

[141] Talbi, E.-G., Roux, O., Fonlupt, C., and Robillard, D. Parallel Ant Colonies for Combinatorial Optimization Problems. In José Rolim et al., editor, *11th IPPS/SPDP'99 Workshops Held in Conjunction with the 13th International Parallel Processing Symposium and 10th Symposium on Parallel and Distributed Processing, April 12-16, San Juan, Puerto Rico*, volume 1586 of *Lecture Notes in Computer Science*, pages 239–247. Springer-Verlag, Berlin, 1999.

[142] ten Eikelder, H.M.M., Aarts, B.J.M., Verhoeven, M.G.A., and Aarts, E.H.L. Sequential and Parallel Local Search for Job Shop Scheduling. In S. Voß, S. Martello, C. Roucairol, and Osman, I.H., editors, *Meta-Heuristics 98: Theory & Applications*, pages 359–371. Kluwer Academic Publishers, Norwell, MA, 1999.

[143] Tongcheng, G. and Chundi, M. Radio Network Design Using Coarse-Grained Parallel Genetic Algorithms with Different Neighbor Topology. In *Proceedings of the 4th World Congress on Intelligent Control and Automation*, volume 3, pages 1840–1843, 2002.

[144] Toulouse, M., Crainic, T.G., and Gendreau, M. 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 Academic Publishers, Norwell, MA, 1996.

[145] Toulouse, M., Crainic, T.G., and Sansó, B. An Experimental Study of Systemic Behavior of Cooperative Search Algorithms. In S. Voß, S. Martello, C. Roucairol, and Osman, I.H., editors, *Meta-Heuristics 98: Theory & Applications*, pages 373–392. Kluwer Academic Publishers, Norwell, MA, 1999.

[146] Toulouse, M., Crainic, T.G., and Sansó, B. Systemic Behavior of Cooperative Search Algorithms. *Parallel Computing*, 30(1):57–79, 2004.

[147] Toulouse, M., Crainic, T.G., Sansó, B., and Thulasiraman, K. Self-Organization in Cooperative Search Algorithms. In *Proceedings of the 1998 IEEE International Conference on Systems, Man, and Cybernetics*, pages 2379–2385. Omnipress, Madisson, WI, 1998.

[148] Toulouse, M., Crainic, T.G., and Thulasiraman, K. Global Optimization Properties of Parallel Cooperative Search Algorithms: A Simulation Study. *Parallel Computing*, 26(1):91–112, 2000.

[149] Toulouse, M., Thulasiraman, K., and Glover, F. Multi-Level Cooperative Search: A New Paradigm for Combinatorial Optimization and an Application to Graph Partitioning. In P. Amestoy, P. Berger, M. Daydé, I. Duff, V. Frayssé, L. Giraud, and D. Ruiz, editors, *5th*

International Euro-Par Parallel Processing Conference, volume 1685 of *Lecture Notes in Computer Science*, pages 533–542. Springer-Verlag, Heidelberg, 1999.

[150] Verhoeven, M.G.A. and Aarts, E.H.L. Parallel Local Search. *Journal of Heuristics*, 1(1):43–65, 1995.

[151] Voß, S. 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.

[152] Wilkerson, R. and Nemer-Preece, N. Parallel Genetic Algorithm to Solve the Satisfiability Problem. In *Proceedings of the 1998 ACM symposium on Applied Computing*, pages 23–28. ACM Press, 1998.