We are IntechOpen, the world's leading publisher of Open Access books Built by scientists, for scientists

6,300

Open access books available

172,000

International authors and editors

190M

Downloads

Our authors are among the

154

Countries delivered to

TOP 1%

most cited scientists

12.2%

Contributors from top 500 universities

WEB OF SCIENCE™

Selection of our books indexed in the Book Citation Index
in Web of Science™ Core Collection (BKCI)

Interested in publishing with us?
Contact book.department@intechopen.com

Numbers displayed above are based on latest data collected.
For more information visit www.intechopen.com

Cooperation Control in Distributed Population-based Algorithms using a Multi-agent Approach Application to a real-life Vehicle Routing Problem

Kamel Belkhelladi, Pierre Chauvet and Arnaud Schaal

LISA, Université d'Angers

IMA, Université Catholique de l'Ouest

France

1. Introduction

Combinatorial optimization problems in transportation, logistic management and other areas involve finding optimal solutions from a discrete set of feasible solutions. However, even with the advent of new computer technologies and parallel processing, many of these problems could not be solved to optimality in reasonable computation times, due to their inner nature or to their size. Moreover, reaching optimal solutions is meaningless in many practical situations, since we are often dealing with rough simplifications of reality and the available data is not precise. The goal of approximate algorithms (or heuristics) is to quickly produce good approximate solutions, without necessarily providing any guarantee of solution quality.

Metaheuristics are general high-level procedures that coordinate simple heuristics and rules to find good (often optimal) approximate solutions to computationally difficult combinatorial optimization problems. Among them, are simulated annealing (Metropolis et al., 1953), tabu search (Glover, 1989; Glover, 1990), Greedy Randomized Adaptive Search Procedure (GRASP) (Feo & Resende, 1989), genetic algorithms (Holland, 1975), scatter search (Laguna & Martí, 2003), Variable Neighbourhood Search (VNS) (Mladenović & Hansen, 1997), ant colonies (Dorigo & Di Caro, 1999), and others. They are based on distinct paradigms and offer different mechanisms to escape from locally optimal solutions, contrarily to greedy algorithms or local search methods. Metaheuristics are among the most effective solution strategies for solving combinatorial optimization problems in practice and they have been applied to a very large variety of areas and situations. The customization (or instantiation) of a metaheuristic to a given problem yields a heuristic to the latter.

Metaheuristics offer a wide range of possibilities for effective parallel algorithms running in much smaller computation times, but requiring efficient implementations. (Cung et al., 2002) showed that parallel implementations of metaheuristics appear quite naturally as an effective approach to speedup the search for good solutions to optimization problems. They allow solving larger problems and finding better solutions with respect to their sequential

counterparts. They also lead to more robust algorithms and this is often reported as the main advantage obtained with parallel implementations of metaheuristics: they are less dependent on parameter tuning and their success is not limited to few or small classes of problems.

Distributed computing refers to computing that involves multiple loosely coupled processors working together to solve an overall problem (Steen & Tanenbaum, 2002). Distributed computing offers a natural approach to solving complex data and computation intensive problems that arise in industry and control. Many approaches to distributed computing have been developed over the past decades. These include Socket Programming, Remote Procedure Calls (RPC), object-oriented DCE, DCOM, CORBA, Java RMI, and Message-Oriented Middleware (MOM) (Steen & Tanenbaum, 2002). These approaches have their own advantages and disadvantages (Steen & Tanenbaum, 2002). Some major limitations with respect to the design of robust, networked applications that involve many autonomous, heterogeneous entities are: (1). Interactions among participating entities, which are fixed a priori through explicitly coded instructions by the application developer. As a result, they lack run-time adaptive behaviour. (2). Ongoing interaction that requires ongoing communication, making it unsuitable for applications that have to operate in environments where maintaining continuous communication is expensive or unfeasible and connections are unreliable. These considerations have motivated the development of approaches to distributed computing based on agents which provide ways of maintaining ongoing interaction without ongoing communication (White, 1997). Multi-agent systems (MAS) and related technologies offer an attractive paradigm for designing distributed networked applications that involve many relatively autonomous, heterogeneous entities (Honavar et al., 1998).

In recent years, the number of strategies proposed to solve complex optimization problems has increased considerably. A new set of problem and model-independent nature inspired heuristic optimization algorithms were proposed by researchers to overcome drawbacks of the classical optimization procedures. These techniques are efficient and flexible. They can be modified and/or adapted to suit specific problem requirements (see Figure 1). Research on these techniques is still continuing all around the globe.

Fig. 1. A pictorial comparison of classical and modern heuristic optimisation strategies

(Adapted from (Baykasoğlu, 2001))

Population-based strategies are very frequently used in metaheuristics (Blum & Roli, 2003; Krasnogor & Smith, 2005). In addition, the complexity of these methods is even higher because they are time-consuming. In this context, the use of parallel and distributed strategies becomes a powerful tool to obtain satisfactory solutions in acceptable runtimes. Though parallel computers can dramatically speed up algorithms, they are not easily available and it is expensive to upgrade their processing power and memory. The expensive parallel computers cannot be expected especially in company-based computing environments. A promising alternative without expensive hardware facilities is to construct a parallel framework on a set of networked personal computers.

In this chapter, we propose a scalable framework for the development of parallel/distributed population-based algorithms using mobile agents launched on different hosts of available networked PCs and cooperating among them to solve large combinatorial problems efficiently. The mobile agent execution environment used to realize our framework is based on the JADE technology (Bellifemine et al., 2001). In fact, parallel/distributed computing applied to problem solving means that several processes work simultaneously on several processors with the common goal of solving a given problem instance. One then has to determine how the global problem solving process is controlled and how information is exchanged among the various processes. Hence, we define a new information exchange strategy based on dynamic migration window methods (Kim, 2002) that control the size and frequency of exchange and a selective migration model (Eldos, 2006) for the choice of information to be exchanged. The proposed framework has been experimented on an extended set of Capacitated Arc Routing Problem (CARP) and a real-life application (Underground Waste Collection Optimization Problem (UWCOP)). Several experiments are carried out on different computer networks of different sizes. Results obtained show the advantages and efficiency of our approach.

The remaining sections of this chapter are organized as follows. In section 2, we overview fundamentals of parallel/distributed population-based algorithms and more specifically parallel/distributed Evolutionary Algorithms models and the potential of multi-agent systems (MAS). Section 3 presents our mobile agent-based computing approach. The MAF-DISTA implementation is described in section 4. In section 5, an NP-hard problem, the Underground Waste Collection Optimization Problem (UWCOP), which we use as a test bed for this study, is presented. Section 6 presents a MAF-DISTA implementation for UWCOP. In section 7, different experiments and results are summarized and discussed. The remaining section presents the conclusion and outlines areas for future research.

2. Background

2.1 An overview on Parallel and Distributed population-based Algorithms

Population-based methods are inspired by natural evolution processes. They handle a population of individuals that evolves with information exchange procedures. Population-based algorithms need constructive method to generate the initial population, and any local search technique can be used to improve each solution in the population. But population-based methods combine good solutions in order to possibly get better ones. The basic idea is that good solutions often share parts with optimal solutions (Hertz & Kobler, 2000).

Parallel and distributed computing may be considered as a mechanism to speed up the search process when solving large optimization problems. Furthermore, the simultaneous use of parallelism and cooperation allows the improvement of the quality of non-dominated solution sets. Evolutionary Algorithms (EAs) are population-based algorithms and very suitable for parallelization, due to the fact that mutation and evaluation can be performed independently on each individual of the population. However, in the selection and crossover processes it is necessary to compare or combine individuals which may be handled on different processors. This implies communications among processors. Parallelization of EAs has been properly studied in the last decade (Cantü-Paz, 2000; Veldhuizen et al., 2003). Most important parallel models are:

- The *master-worker* model, which allows keeping the sequentiality of the original algorithm. The *master* centralizes the population and manages the selection and replacement steps. It sends subpopulations to *workers* that execute recombination and evaluation tasks. The *workers* return newly evaluated solutions back to the *master*. This approach is efficient when the cost of generating and evaluating new solutions is high. In some cases the island model improves the quality of the solutions reached because the islands help to maintain diversity.
- The *island* or *coarse-grain* model, which divides the entire population into several subpopulations distributed among different processors. Each processor is responsible for the evolution of one subpopulation. It executes all the steps of the metaheuristic, and occasionally individuals migrate among islands. The goal is to obtain solutions for at least the same quality than the *master-worker* model, but in a shorter runtime.
- The *diffusion* or *fine-grain* model, which works with a population. Each processor is responsible for either a single individual or at most a small number. The difference with respect to the island paradigm is that the diffusion-based scheme requires a neighbourhood structure of processors to perform the recombination and selection.

2.2 Potential of multi-agent systems

Multi-agent systems (MAS) have proven to be an effective paradigm in a number of distributed networked applications that require information integration from multiple heterogeneous autonomous entities (Caragea et al., 2001). A multi-agent system consisting of multiple agents can take advantage of computational resources and capabilities that are distributed across a network of interconnected entities. An agent-based approach allows the creation of systems that are flexible, robust, and can adapt to the environment. This is especially helpful when components of the system are not known in advance, change over time, and are highly heterogeneous. Agent-based computing offers the ability to decentralize computing solutions by incorporating autonomy and intelligence into cooperative, distributed applications. Each agent perceives (the state of its environment), infers (updates its internal knowledge according to the newly received perceptions), decides (on an action), and acts (to change the state of the environment). Agent-oriented programming (AOP) is the software paradigm used to facilitate agent-based computing and extends from object-oriented programming (OOP) by replacing the notions of class and inheritance with the notions of roles and messages, respectively (Lind, 2001). If one wants to develop useful and applicable multi-agent system development tools, several characteristics are essential in practice. Among these characteristics, one can cite:

- Simple support for the deployment of inter-computer and distributed applications.

- Significant reduction of the effort required implementing a MAS, especially when considering the important amount of programming usually involved.
- Effective and efficient abstraction in order to allow people who are less experienced with agent-oriented programming to relatively easily create MAS without in-depth knowledge of all implementation details.
- Sufficient latitude to most experienced users, who then can get access to and interact directly with various system components; the tool should add an abstraction level to programming without becoming an obstacle for programmers.
- Implementation (code) is easily extensible.
- Developers do not have to worry about system communications implementation and associated protocols used to transfer messages among agents.
- Availability of a debug utility, of a user interface facilitating development, and of an automatic source code generator.
- The environment is supported by a suitable documentation.

(Garneau & Delisle, 2003) have done a rigorous study to evaluate and compare several MAS environments in order to identify tools' and environments' strong points and weak points in the aim of determining what a complete development environment should look like. Table 1 summarizes some results obtained by (Garneau & Delisle, 2003). Eight agent-oriented programming tools have been chosen because of their popularity and their relevance: Jade (Bellifemine et al., 2001), Zeus (Lee et al., 1998), Madkit (Gutknecht & Ferber, 2001), AgentBuilder (AgentBuilder, 2004), Jack (Busetta et al., 1999), Decaf (Graham et al., 2000), JafMas/Jive (Chauhan, 1997) and AgentTool (Deloach & Wood, 2000).

Mobile agents are a convenient paradigm for distributed computing (Bic et al., 1996) (Chess et al., 1996). The agent specifies when and where to migrate, and the system handles the transmission. This makes mobile agents easier to use than low-level facilities in which the programmer must explicitly handle communication, but more flexible and powerful than schemes such as process migration in which the system decides when to move a program based on a small set of fixed criteria. Mobile agents allow a distributed application to be written as a single program.

As we are interested in JADE platform, we report a short description in the next subsection.

2.2.1 JADE

Java Agent DEvelopment Framework (JADE) middleware (JADE-Tilab, 2009) is a Java agent platform which provides a fully distributed system, where agents can reside in different hosts, an efficient transport mechanism with asynchronous messages, and internal (within a single JADE middleware platform instance) agent mobility support. An important feature of JADE is support for ontologies and content languages used in the agent messages. Each JADE agent runs as an independent thread and executes behaviours. Behaviours are tasks that can be dynamically inserted into and scheduled by an agent.

JADE is designed to ease the development of multi-agent systems by providing standard agent services and methods. It also enables interoperability among different agent platforms since it complies with *FIPA* standards. The Jade platform is composed of one *main-container* and several *containers* that can be launched on one or more hosts distributed across the network. Each *container* can run zero, one or multiple agents. The following agents and services must be present in the *main-container* at any times: the AMS (Agent Management System), the DF (Directory Facilitator) and the MTS (Message Transport System) which is

also referred to as the ACC (Agent Communication Channel). The JADE middleware architecture is depicted below in figure 2.

	JADE	DECAF	AgentBuilder	Zeus	JASMAF/JIVE	Jack	AgentTool	Madkit
Evaluation criteria								
Methodology	0	0	4	4	3	0	3	3
Learning facility	0	3	1	1	1	0	3	2
Step transition	0	0	3	3	2	0	3	2
Tool flexibility	3	0	1	1	2	3	0	3
Communication	4	2	4	4	2	3	2	3
Debug utility	3	2	4	4	1	0	2	4
Development support	0	2	4	4	2	1	4	2
Implementation support	0	0	4	4	2	1	2	1
MAS management	4	0	3	3	0	0	1	4
Implementation simplicity	2	3	2	2	1	2	3	1
Code generation	0	0	1	3	1	0	1	0
Code extensibility	4	1	1	2	1	4	0	3
Deployment	4	1	2	2	1	2	1	3
Documentation	3	1	4	4	1	3	1	3
Total	27	15	38	41	20	19	26	34

Table 1. Grid results

Fig. 2. The JADE agent middleware architecture (WIDE, 2009)

3. A Mobile Agent framework as a tool for implementing parallel Population-based algorithms (MAF-DISTA)

Implementing parallel EAs involves expensive parallel computers that normally cannot be expected in a company computing environment. A promising alternative without expensive hardware facilities is to construct a parallel computational framework on a set of networked computers. There are some possible ways to manage the operation of such a distributed computational framework, for example a *client-server* technique or an agent-based approach. *Client-server* is the most common paradigm of distributed computing at present. However, in this paradigm all components are stationary with respect to execution. This makes it unsuitable for the goal here no-specific computational resources are preserved and dedicated to the parallel EAs so that the EC-code needs to be moved occasionally to the available machines. Under such circumstances, mobile agent-based design paradigm provides a better choice to support parallel computing (Lange & Oshima, 1999; Macêdo & Silva, 2005). Mobile agents are software agents that are capable of being transmitted themselves (their program and their state) across a computer network and recommencing execution at a remote site. They have been proposed to solve problems of networked application domains, for example, electronic commerce (Dasgupta et al., 1999; Leeand et al., 2003), Network Management (Papavassiliou et al., 2002; Jiang et al., 2005), and information retrieval (Thati et al., 2001). Many different multi-agent platforms are publicly available, including D'Agent (Gray et al., 2002), Tracy (Braun & Rossak, 2005), Aglets (Lange & Oshima, 1998) and JADE (Bellifemine et al., 2001). Application developers can use them as platforms and focus on the software development issues at application level. In this work, we chose to use the JADE toolkit and take the mobile agent approach to develop an adaptive computational model to parallelize EAs. We implemented a *coarse-grain* population framework on a set of networked computers. This network of computers is similar to a DM-MIMD (Distributed Memory Multiple Instruction stream, Multiple Data stream) computer and its increasing availability makes it an ideal and practical platform. Figure 3 presents our framework architecture.

Fig. 3. MAF-DISTA architecture

We used Java mobile agents of the JADE framework, which break the barriers of heterogeneous environments. JADE is compliant with FIPA standard specifications, and

agents developed with it can thus interoperate with other agents built with the same standard (O'Brien & Nicol, 1998). From the functional point of view, JADE provides the basic services necessary for distributed peer-to-peer applications in a networked environment. It supports mobility of code and execution state: an agent can stop its execution on a host, migrate to another host, and then restarts its execution. In this environment, JADE allows each agent to dynamically discover other agents and to communicate with them in a peer-to-peer manner.

4. MAF-DISTA implementation

We defined five main kinds of agents (in addition to the default agents provided by JADE for network services): the mobile agent (MA) to carry out Evolutionary Computation (EC), the main status agent (MSA) and the local status agent (LSA) to report machine status, the synchronization agent (SYNCA) to summarize the evolving progress of subpopulations and the user interface agent (UIA) to publish computing results and allow users to interact with the computing system. In the proposed framework, mobile agents have the ability to migrate from machine to machine. In our strategy, each mobile agent runs a subpopulation. Initially, a local status agent (LSA) is created for each machine in the network framework, and the main status agent (MSA) in the main host is responsible for maintaining the information of individual machine status (e.g., the CPU utility) reported by other status agents. Then, the mobile agent (MA) starts computation in the main host; it clones itself using the EC-code for each node of the prespecified n-cube model. According to the information provided by MSA, MA dispatches the duplication to each available machine. Therefore, during the period of execution if a status agent in any machine (except the main host) detects the existence of a new end-user and the CPU utility exceeds a certain *threshold*¹, the mobile agent in the same machine is informed to interrupt the execution of its corresponding EC-code, in order to release the computing resources for the end-user. The mobile agent then inquires machine availability from LSA, to resume the execution. If the mobile agent finds a machine whose CPU utility is lower than the *threshold*, it asks MSA to reserve this machine, and then carries its code and related information to that machine to continue the computation, starting from the point where it has been interrupted; otherwise it stays in the machine it has been in and waits for any available one. The implementation of the proposed approach is more detailed in the sequence diagram of figure 4.

¹ Percentage of CPU Utility

IntechOpen

IntechOpen

Fig. 4. MAF-DISTA Sequence Diagram

4.1 Cooperation control

The goal of maintaining genetic diversity is to prevent *premature convergence*. Separate subpopulations could have their respective genetic features. Migrations among subpopulations are necessary to help these small populations to jump out of local minima. Combinations of good genes from different species are expected to result in even better offspring. Therefore, the best individuals are the candidates to be copied to other environments. Since the clones of the global champion may dominate all subpopulations and reduce the genetic diversity, migration should be carefully directed. Figure 5 shows the aspect of our computational framework; it illustrates data and control flows of the computation within a migration interval (i.e., consecutive k generations). In this figure, each grey block contains a subpopulation of individuals in which evolutionary computation continues for a certain number of generations before migration happens. During this period, computation for each subpopulation is independent from others, so evolution for different subpopulations can proceed simultaneously. In our work, we developed a new information exchange strategy based on dynamic migration window methods (Kim, 2002) that control the size and frequency of exchanges and a selective migration model (Eldos, 2006) for the choice of information to be exchanged.

Fig. 5. MAF-DISTA cooperation control

In the proposed model, individuals are examined at both the source and destination levels, to qualify for migration. The source gives or denies a visa based on local qualification criteria and the destination grants or denies a residency based on local qualification criteria. The simplest form of qualification criteria is based on individuals' relative fitness. Every

individual in a subpopulation is ranked locally; an individual qualifies for a visa at the source subpopulation if its rank is within a range, typically the middle class. In the destination subpopulation, an immigrant is accepted as a new member of the population if its rank is better than a *threshold*² set by the receiving subpopulation.

5. Case study in optimization problems

The proposed approach has been experimented on an industrial problem. In this section, we focused on a real-life application (Underground Waste Collection Optimization Problem).

5.1 Problem description

The underground waste storage system in Angers (France) is an innovative way of containing waste. This system consists of two main components, a 4 or 5 m^3 underground waste container with a specially designed deposit point, and a collection vehicle specifically developed and featuring a side-loading system that enables a single driver/operator to empty containers using a fully automated process. The total collection cycle for a container takes approximately 5 minutes from the moment the collecting vehicle arrives to the time it is ready to move off again. The collecting vehicle's 20 m^3 compacting container offers an equivalent load capacity of about 9.5 tons. As in other French cities, most of the population of Angers is concentrated in urban areas with a very high population density, given that people usually live in apartment buildings. As a result, the collection technique has evolved away from door-to-door collection, and the predominant system consists of collection points located throughout the city where citizens leave their refuse. Each collection point is made up of one or more underground waste containers. Different trucks collect each type of waste and transport it to its final destination (a refuse dump, an incinerator or a recycling plant). One key factor for proper operation of collection systems is the design of appropriate collection routes for each type of garbage (fraction), which may or may not be separately collected (in some cases due to technical design constraints and different treatment processes). In Angers each fraction is collected separately with a fixed periodicity (established by the municipality). Therefore, in this case the collection of each type of garbage (urban refuse, generic waste materials like paper, plastics, packaging, glass, etc.) can be considered as an independent but equivalent problem. In addition, the routes should be coherent with traffic rules, taking into account prohibited turns and traffic signals while minimizing collection costs. A container must be collected if its filling rate is more than a *threshold*³ (20% of the container size, in our case). Some containers are "black spots" and must be collected even if their content is lower than the *threshold*. These containers are located in sensitive places, generally public structures such as markets, prisons, etc. Finally, two distinct costs per required route are considered, *deadheading* and *collecting*.

5.2 Problem modeling and formulation

UWCOP is an arc routing problem since roads are completely traversed, even if collection points are located in specific spots of the roads. In addition, the capacity of trucks is limited.

² Qualification criterion

³ Percentage of container size

Thus, the basic nature of UWCOP is that of a Capacitated Arc Routing Problem (CARP) that can be defined as follows. Suppose a connected undirected graph $G=(V,E)$, deadheading costs $dc:E \rightarrow \mathbb{Z}^+$, collecting costs $cc:E \rightarrow \mathbb{Z}^+$, demands $w:E \rightarrow \mathbb{Z}^+$, a fleet of K identical vehicles with capacity W is based at a depot node s . Define $R = \{(i,j) \in E \mid w((i,j)) > 0\}$ as the set of required edges. Let F be a set of closed walks that start and end at the depot, where edges in a walk can be either *served* or *deadheaded*. Set F is a feasible CARP solution if:

- Each required edge is serviced by exactly one walk in F ;
- The sum of demands of the serviced edges in each walk in F does not exceed the vehicle capacity W .

We want to find a solution minimizing the total costs of the walks. It can be noted that $\sum_{(i,j) \in R} dc(i,j) + \sum_{(i,j) \in R} cc(i,j)$ is a lower bound on the cost of an optimal solution, the remaining costs in a solution are the costs of the deadheaded edges.

The UWCOP can be described by the following programming problem:

Notations:

E : Set of network edges.

R : Set of network edges (i,j) whose demands are greater than zero.

V : Set of network nodes.

I : Set of vehicles.

W : Vehicle capacity.

K : Number of vehicles.

C : Container capacity.

Q_c : Quantity of waste contained in container c .

M_{ij} : Number of underground containers placed on edge (i,j) .

T_{cc} : A container total collection cycle.

v_c : Denotes a binary variable equal to 1 if $Q_c \geq 20\% \times C$ and 0 otherwise.

b_c : Denotes a binary variable equal to 1 if the container c is a black spot and 0 otherwise.

dc_{ij} : Deadheading cost of edge (i,j) .

cc_{ij} : Collecting cost of edge (i,j) .

w_{ij} : Demand of edge (i,j) or the total demands between nodes i and j .

x_{ijk} : Denotes a binary variable equal to 1 if the vehicle k traverses the edge (i,j) from i to j and 0 otherwise.

l_{ijk} : Denotes a binary variable equal to 1 if the edge (i,j) is serviced by the vehicle k from i to j and 0 otherwise.

$$\text{Minimize } \sum_{k \in I} \sum_{(i,j) \in E} dc_{ij} x_{ijk} + \sum_{(i,j) \in R} cc_{ij} \quad (1)$$

$$\forall i \in V, \forall k \in I: \sum_{j \in \Gamma(i)} (x_{ijk} - x_{jik}) = 0 \quad (2)$$

$$\forall (i,j) \in E, \forall k \in I: x_{ijk} \geq l_{ijk} \quad (3)$$

$$\forall (i,j) \in R: \sum_{k \in I} (l_{ijk} + l_{jik}) = 1 \quad (4)$$

$$\forall k \in I: \sum_{(i,j) \in R} l_{ijk} w_{ij} \leq W \quad (5)$$

$$\forall (i,j) \in R, cc_{ij} = \sum_{m=1}^{M_{ij}} T_{cc} \times (v_c \vee b_c) \quad (6)$$

$$u_s^k, y_s^k, x_{ijk}, l_{ijk}, v_c, b_c \in \{0, 1\} \quad (7)$$

$$\forall S \neq \emptyset, S \subset \{2, \dots, n\}, \forall k \in I : \begin{cases} \sum_{i,j \in S} x_{ijk} - n^2 y_s^k \leq |S| - 1 \\ \sum_{i \in S} \sum_{j \in S} x_{ijk} + u_s^k \geq 1 \\ u_s^k + y_s^k \leq 1 \end{cases} \quad (8)$$

The goal (1) is to determine a set of closed walks (vehicle trips) of minimum total cost. Constraint (2) ensures that a vehicle coming into a node must leave it again. Constraint (3) ensures that a required edge is also traversed. Each required edge is serviced by one single trip (4) such that each trip starts and ends at the depot, and the total demand handled by any vehicle does not exceed W (5). A container must be collected if its filling rate is more than 20% or it is a *black spot* (6). Finally, illegal sub-trips are forbidden by constraint (8) which requires additional binary variables such as y_s^k and u_s^k .

Since the CARP is NP-hard, large scale instances must be solved in practice using heuristics. Among fast constructive methods, one can cite Path-Scanning (Golden et al., 1983), Augment-Merge (Golden & Wong, 1981) and Ulusoy's splitting technique (Ulusoy, 1985). Available metaheuristics are very recent and include tabu search methods (Hertz et al., 2000; Belenguer & Benavent, 2003), guided local search (Beullens et al., 2003) and memetic algorithms (Lacomme et al., 2001; Lacomme et al., 2004). All these heuristics algorithms can be evaluated through lower bounds (Amberg & Voß, 2002).

6. MAF-DISTA implementation for UWCOP

We conducted two series of experiments to compare the corresponding performance for evolutionary computation with and without parallelism. In these experiments, each mobile agent executes a conventional genetic algorithm. The genetic algorithm (GA) developed in this work uses several components of the genetic algorithm and the effective memetic algorithm proposed by (Lacomme et al., 2001; Lacomme et al., 2004) for the CARP. The common parts are described below:

6.1 Solution encoding

The network is coded as a symmetric digraph, in which each edge is replaced by two opposite arcs. A chromosome is an ordered list of the T tasks, in which each task may appear as one of two directions. The implicit shortest paths are assumed between successive tasks. The chromosome does not include trip delimiters and can be viewed as a *full trip* for an incapacitated vehicle. A *Split* procedure optimally partitions (subject to the sequence) the *full trip* into feasible trips. This procedure is inspired by the Ulusoy's splitting technique (Ulusoy, 1985). Usually, this technique is a CARP heuristic, which is better explained in figure 6. First, capacity is ignored to build one *full trip* τ covering all tasks (a, b, c, d and e in the figure). Second, an auxiliary graph is built, in which each arc denotes a subsequence of τ that can be done by one trip. Each arc is weighted by the cost of this trip. A shortest path in this graph shows where τ should be split into trips, and gives the cost of the corresponding solution. Third, the solution is built with one trip per arc on this path. The fitness function of the GA is the total cost of the resulting CARP solution.

Fig. 6. Ulusoy's splitting technique

6.2 Initialization

The global population P of chromosomes is initialized with the solutions of the three CARP heuristics (PS, AM and UH) (Golden & Wong, 1981; Golden et al., 1983; Ulusoy, 1985), completed by random permutations. Clones (identical chromosomes) are forbidden.

6.3 Selection and crossover

At each iteration, two parents are selected by a biased roulette wheel (Goldberg, 1989). The crossover operator used in the algorithm consists in applying *Order Crossover* (OX), *Linear Order Crossover* (LOX) or *One point Crossover* (X1) operations with equal probability (see figure 7). *Order Crossover* tends to transmit the relative positions of genes rather than the absolute ones. In the OX, the chromosome is considered to be circular. LOX is a modified version of *Order Crossover*, where the chromosome is considered linear instead of circular. The LOX works as follows:

- Select sub-lists from parents (p_1 , p_2) randomly.
- Remove $sublist_2$ from parent p_1 leaving some "holes" and then slide the holes from the extremities toward the center until they reach the cross section. Similarly, remove $sublist_1$ from parent p_2 and slide holes to cross section.
- Insert $sublist_1$ into the holes of parent p_2 to form the offspring o_1 and insert $sublist_2$ into the holes of parent p_1 to form the offspring o_2

Fig. 7. Crossovers

6.4 Mutation

The mutation operator used in the algorithm consists in applying *displacement* (MOVE) or *reciprocal exchange* (SWAP) operations to each gene with equal probability (see figure 8). The *displacement* mutation selects two positions p and q in the chromosome such as $1 \leq p \leq T, 0 \leq q \leq T, p \neq q$ and $p \neq q - 1$. Task u in rank p is moved after that of rank q ($q=0$ means an insertion in the lead). *Reciprocal exchange* mutation selects two positions at random and swaps the tasks on these positions.

Fig. 8. Mutations

7. Experimental results and discussion

The experiment had a dual objective. On the one hand, we aimed to compare the cooperative search with the simple version and with the best performing methods proposed in the literature dealing with CARP and, thus, to validate our claim that the proposed method offers competitive performance in terms of both solution quality and computational effort. On the other hand, we also aimed to evaluate the influence of subpopulations number and how the proposed mobile agent approach could support adaptive parallelism on networked computers.

In all series of experiments, we used three crossover types (LOX, OX, X1) and two mutation types (MOVE, SWAP). Ten values for crossover rate were used ranging from 0.1 to 0.99 in increments of 0.1. Also, ten mutation rates were allowed varying from 0.1 to 0.55 in increments of 0.05. Clones (identical solutions) were forbidden in each subpopulation, to

have a better dispersion of solutions and to diminish the risk of *premature convergence*. The number of generations was fixed to 5000 for all mobile agents. The migration interval was 500 generations. Whenever migration occurred, the dynamic migration window size varied at random from 1 to θ . The θ value was generated at random within 20% of the subpopulation size. The *threshold* value was fixed to the mean fitness of the subpopulation. Furthermore, 4 runs are carried out in each experiment.

These tests were done on a standard set of undirected instances in which all edges were required. Table 2 contained 34 instances from (Belenguer & Benavent, 2003) with 24 to 50 nodes and 34 to 97 edges. All these files could be obtained at <http://www.uv.es/belengue/carp.html>. We modified these files by adding a deadheading and collecting costs. Thus, we applied our framework on an instance of UWCOP based on real data provided by the Waste Management and Environment Agency of Angers, France. Results are shown in table 3.

In all tables, *PB* gives the instance number and *N*, *M* the numbers of nodes and edges. *CC* gives the collecting cost. In table 2, *LBB* is a lower bound from (Belenguer & Benavent, 2003). *TS* is the result of Carpet (Hertz et al., 2000) with the parameter setting yielding the best results on average (the same setting for all instances). *BEST* gives the best solution published, generally obtained by Carpet with various parameter settings. *MA* is the solution of the competitive memetic algorithm from (Lacomme et al., 2004).

Our results are shown in the Static Parallel Algorithm (*SPA*) without individual's exchange and the distributed genetic algorithm including individual exchange strategy (*DAIE*) columns.

We compared our algorithms with the best techniques so far reported in the literature. We have selected some newer algorithms, like lower bound proposed by (Belenguer & Benavent, 2003), tabu search (Hertz et al., 2000) and the competitive memetic algorithm from (Lacomme et al., 2004). The last two techniques are known to perform very well on the CARP.

7.1 Results on Belenguer and Benavent's instances

In 90% of the 34 instances of (Belenguer & Benavent, 2003), our *DAIE* outperformed *SPA* implementation of evolutionary system by a mean advantage of 2.62% in term of solution cost. *DAIE* was very efficient: on all instances, it was at least as good as Carpet. On the 34 instances, it outperformed Carpet 17 times, improved 9 best known solutions with 6 to optimality, and reached *LBB* 27 times. The average deviation to *LBB* was roughly divided by 5 compared to Carpet, and reached 0.25%. Moreover, on the same instances, *DAIE* outperformed the Competitive Memetic Algorithm (Lacomme et al., 2004) 7 times.

PB	N	M	CC	LBB	BEST	TS	MA	SPA	DAIE
1A	24	39	195	368	368	368	368	368	368
1B	24	39	195	368	368	368	368	376	368
1C	24	39	195	430	440	440	440	445	430
2A	24	34	170	397	397	397	397	405	397
2B	24	34	170	429	429	430	429	440	429
2C	24	34	170	625	627	664	632	649	625
3A	24	35	175	256	256	256	256	257	256
3B	24	35	175	262	262	262	262	264	262
3C	24	35	175	312	313	313	313	319	312
4A	41	69	345	745	745	745	745	786	745
4B	41	69	345	757	757	761	757	775	761
4C	41	69	345	773	773	798	773	782	773
4D	41	69	345	865	886	901	886	940	881
5A	34	65	325	748	748	748	748	748	748
5B	34	65	325	771	771	773	771	773	771
5C	34	65	325	794	799	801	799	849	801
5D	34	65	325	896	906	932	906	932	902
6A	31	50	250	473	473	473	473	495	473
6B	31	50	250	481	483	491	483	495	481
6C	31	50	250	561	567	579	567	582	567
7A	40	66	330	609	609	609	609	609	609
7B	40	66	330	613	613	613	613	636	613
7C	40	66	330	663	664	673	664	673	663
8A	30	63	315	701	701	701	701	709	701
8B	30	63	315	710	710	716	710	716	710
8C	30	63	315	832	843	848	848	907	832
9A	50	92	460	783	783	783	783	830	783
9B	50	92	460	786	786	789	786	796	786
9C	50	92	460	792	792	792	792	798	792
9D	50	92	460	842	851	869	851	873	851
10A	50	97	485	913	913	913	913	915	913
10B	50	97	485	921	921	921	921	925	921
10C	50	97	485	931	931	936	931	943	931
10D	50	97	485	1009	1020	1029	1020	1045	1015

Table 2. Computational results for instances of (Belenguer & Benavent, 2003)

7.2 Results on Angers’s instance

Table 3 shows the results for the file we constructed from real data provided by the Angers Waste Management and Environment Agency. On this instance, the *DAIE* outperforms the *SPA* by a mean advantage of 3% in term of solution cost.

PB	N	M	CC	SPA	DAIE
Angers1	64	67	585	920	892

Table 3. Computational results for UWCOP in Angers

7.3 The influence of subpopulation number on solution quality (Fitness)

The number of subpopulations determines the size of subpopulation because the original population is distributed among subpopulations. The greater the number of subpopulations is high, the more the size of each subpopulation is reduced (Cantü-Paz, 2000). The influence of the number of subpopulations on the quality of solution was investigated. The number of subpopulations varied from 1 to 10. The initial population in every test was the same. In fact, we conducted experiments on one subpopulation of 800 individuals, two subpopulations of 400 individuals, four subpopulations of 200 individuals, six subpopulations of 133 individuals, eight subpopulations of 100 individuals, and ten subpopulations of 80 individuals, in similar experimental conditions, as described above. Each subpopulation was executed on one computer and end-users were not allowed to access the machines used for the experiments. Figure 9 represents the variation of fitness (objective function) compared to the variation of the number of subpopulations. In this figure, one can note that the quality of the solution increases progressively as the number of subpopulations increases. Indeed, the number of subpopulations has a direct influence on the genetic diversity of this parallel model (*DAIE*). A large number of subpopulations implies a reduction of the number of individuals per subpopulation (reduction of the number of genes). As clones are forbidden, subpopulations become heterogeneous and enable interesting subspaces of solutions to be explored.

Fig. 9. Influence of subpopulation number on fitness

7.4 The influence of the number of processors on the speed-up

In the same conditions defined in the previous subsection, the influence of the number of processors on the speed-up of distributed evolutionary algorithm was considered. Figure 10 shows the computational time spent on each experiment. As can be observed, the time for running a single experiment is reduced in an almost linear manner, depending on the number of computers used. Although in the parallel model, agent communication for exchanging individuals among subpopulations needs extra computational effort, and is relatively small compared to the time for running a time-consuming evolutionary experiment.

Fig. 10. Effect of the number of processors on speed-up

7.5 The influence of CPU utility on the speed-up

In this experiment, eleven hosts were connected to the JADE platform in which only one was preserved for running EC experiments. The preserved host was initiated as the main-container to enable the default agents AMS, DF, and RMA, and end-users were allowed to use the other ten computers as they usually did. Each host was a container that included a mobile agent (MA) and a local status agent (LSA).

Due to different user-accessing situations, the experiments were conducted five times in the same day. Figure 11 shows the results. As can be seen, all runs have been sped up using the proposed approach. It shows the efficiency of our mobile agent-based approach. Though the computational cost of adaptive parallelism is higher than that of static parallelism presented above, the former is nevertheless a more practical way to realize parallelism for evolutionary computation in an ordinary networked computing environment.

Fig. 11. Adaptive parallelism using mobile agents (influence of CPU utility on the speed-up)

8. Conclusion and future work

In this chapter, we have proposed MAF-DISTA, an efficient and scalable framework that combines the two advantages of parallelism: computational power and cooperation. We implemented an information exchange strategy based on dynamic migration window methods that control the size and frequency of migration and the selective migration model for the choice of individuals to migrate. Results confirmed the positive impact of using our individual's exchange strategy in regard to the static parallel algorithm. They also pointed out that such strategies are, at a minimum, as good as the best known methods.

Furthermore, the proposed framework was a middleware supporting development of distributed population-based techniques. Its main advantages that have been inherited from JADE, include:

- Ability to simplify the development of distributed population-based metaheuristics composed of autonomous entities that need to communicate and collaborate in order to achieve the working of the entire system.
- Interoperability - mobile agents can cooperate with other agents provided that they comply with the FIPA standard.
- Uniformity and portability - the framework provides a homogeneous set of APIs that are independent from the underlying network and Java version.
- Maximum design and Code reuse - the framework provides the user with a complete architecture design of his/her solution method. Moreover, the programmer may redo as little code as possible. It also provides a maximal conceptual separation between the solution methods and the problem to be solved. The user might therefore develop only the minimal problem-specific code.

According to these features, we can conclude that this middleware is able to be extensible.

In future work, we will implement other population-based metaheuristics on our framework such as ant colony algorithm and particle swarm algorithm. We will also test our framework on other NP-complete problems such as planning and scheduling problems.

9. Acknowledgements

Our research is supported through the municipality of Angers and the Urban Waste Collection and Environment Agency. This support is gratefully acknowledged.

10. References

- AgentBuilder (2004). An Integrated Toolkit for Constructing Intelligent Software Agents, AgentBuilder, Reference Manual, available at <http://www.agentbuilder.com>, June 2004
- Amberg, A. & Voß, S. (2002). A Hierarchical Relaxations Lower Bound for the Capacitated Arc Routing Problem, *Proceedings of Annual Hawaii International Conference on System Sciences*, pp. 1415 – 1424, Hawaii
- Baykasoğlu, A. (2001). Goal Programming using the Multiple Objective Tabu Search. *Journal of Operational Research Society*, 52, 12, 1359-1369
- Belenguer, J.M. & Benavent, E. (2003). A cutting plane algorithm for the capacitated arc routing problem. *Comput. Oper. Res.* 30, 5 (April 2003) 705-728, 0305-0548
- Bellifemine, F.; Poggi, A. & Rimassa, G. (2001). Developing multi-agent systems with a FIPA-compliant agent framework. *Software: Practice and Experience*, 31, 2, 103-128, 0038-0644
- Beullens, P.; Muyldermans, L.; Cattrysse, D. & Oudheusden, D.V. (2003). A guided local search heuristic for the capacitated arc routing problem. *European Journal of Operational Research*, 127, 3 (June 2003) 629-643.
- Bic, L.; Fukuda, M. & Dillencourt, M. (1996). Distributed computing using autonomous objects. *IEEE Computer*, 18, 55-61, 1996. Aug 1996
- Blum, C. & Roli, A. (2003). Metaheuristics in combinatorial optimization: Overview and conceptual comparison. *ACM Computing Surveys*, 35, 3, 268-308, 0360-0300
- Braun, P. & Rossak, W. (2005). Mobile agents: Basic concepts, mobility models, and the tracy toolkit, CA: Morgan Kaufmann Publishers, Los Altos, 2005.
- Busetta, P.; Ronnquist, R.; Hodgson, A. & Lucas, A. (1999) Jack: Intelligent Agents Components for Intelligent Agents in Java, *AgentLink News Letter*, 2, 2-5
- Cantü-Paz, E. (2000). Markov chain models of parallel genetic algorithms. *IEEE Transactions on Evolutionary Computation*, 4, 3, (September 2000) 216-226, 1089-778X
- Caragea, D.; Silvescu, A. & Honavar, V. (2001). Towards a Theoretical Framework for Analysis and Synthesis of Agents That Learn from Distributed Dynamic Data Sources, In: *Emerging Neural Architectures Based on Neuroscience*, 547-559 Springer-Verlag New York, Inc., 3-540-42363-X, New York, USA
- Chauhan, D. (1997). JAFMAS: A Java-Based Agent Framework for Multi-Agent Systems Development and Implementation, *Masters Thesis*, ECECS Department, University of Cincinnati
- Chess, D.; Harrison, C. & Kershenbaum, A. (1996). Mobile Agents: Are They a Good Idea? *Proceedings of the Second International Workshop on Mobile Object Systems*, Jan Vitek,

- Christian Tschudin (Ed.), pp. 25-47, Lecture Notes in Computer Science, Springer-Verlag, Linz, Austria, 1222, July 1996
- Cung, V.-D.; Martins, S.L.; Ribeiro, C.C. & Roucairol, C. (2002). Strategies for the parallel implementation of metaheuristics, In: *Essays and Surveys in Metaheuristics*, Ribeiro, C.C. (Ed.), page numbers (263-308), Kluwer, 978-0792375203, Dordrecht
- Dasgupta, P.; Narasimhan, N.; Moser, L.E. & Melliar-Smith, P.M. (1999). MAgNET: Mobile agents for networked electronic trading. *IEEE Transactions on Knowledge and Data Engineering*, 11, 4 (August 1999) 509-525, 1041-4347
- DeLoach, S.A. & Wood, M. (2000). Developing Multiagent Systems with agentTool, *Proceedings of International Workshop on Agent Theories, Architectures and Languages*, pp. 46-60, 3-540-42422-9, Berlin, Germany
- Dorigo, M. & Di Caro, G. (1999). The Ant Colony Optimization Meta-Heuristic, In: *New Ideas in Optimization*, 11-32, McGraw-Hill, 0-07-709506-5, New York
- Eldos, T. (2006). A New Migration Model For Distributed Genetic Algorithms, *Proceedings of International Conference On Scientific Computing*, pp. 128-134, Las Vegas, NV, USA
- Feo, T.A. & Resende, M. G. C. (1989). A probabilistic heuristic for a computationally difficult set covering problem. *Operations Research Letters*, 8, 2, 67-71, 0167-6377
- Garneau, T. & Delisle, S. (2003). A New General, Flexible and Java-Based Software Development Tool for Multiagent Systems, *Proceedings of International Conference on Information Systems and Engineering*, pp. 22-29, Montréal (Québec, Canada), July 2003
- Glover, F. (1989). Tabu Search – Part I, *ORSA Journal on Computing*, 1, 3, 190-206.
- Glover, F. (1990) Tabu Search – Part II, *ORSA Journal on Computing*, 2, 1, 4-32.
- Goldberg, D. E. (1989). *Genetic Algorithms in Search, Optimization and Machine Learning*, Addison-Wesley Longman Publishing Co., Inc., 0201157675, Boston, MA, USA
- Golden, B.L.; DeArmon, J.S. & Baker, E.K. (1983). Computational Experiments with Algorithms for a Class of Routing Problems. *Computers and Operations Research*, 10, 1, 47-59
- Golden, B.L. & Wong, R.T. (1981). Capacitated Routing Problems. *Networks*, 11, 305-315
- Graham, J.R.; Mchugh, D.; Mersic, M.; McGeary, F.; Windley, M.V.; Cleaver, D. & Decker, K.S. (2000). Tools for Developing and Monitoring Agents in Distributed Multi-Agent Systems, *Proceedings of Agents Workshop on Infrastructure for Multi-Agent Systems*, pp. 12-27, 3-540-42315-X, Barcelona, Spain
- Gray, R.; Cybenko, G.; Kotz, D.; Peterson, R. & Rus, D. (2002). D'Agents: Applications and performance of a mobile-agent system. *Software: Practice and Experience*, 32, 6 (2002) 543-573
- Gutknecht, O. & Ferber, J. (2001). The MADKIT Agent Platform Architecture, In: *Revised Papers from the International Workshop on Infrastructure for Multi-Agent Systems*, 48-55, Springer-Verlag, 3-540-42315-X, London, UK
- Hertz, A. & Kobler, D. (2000). A framework for the description of evolutionary algorithms. *European Journal of Operational Research*, 126, 1, (October 2000) 1-12, 0377-2217
- Hertz, A.; Laporte, G. & Mittaz, M. (2000). A Tabu Search Heuristic for the Capacitated Arc Routing Problem. *Oper. Res.*, 48, 1, 129-135, 0030-364X
- Holland, J. H. (1975). *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence*, The MIT Press, 0262581116, USA

- Honavar, V.; Miller, L. & Wong, J. (1998). Distributed Knowledge Networks, *Proceedings of IEEE Information Technology Conference*, pp. 87-90, 0-7803-9914-5, Syracuse, NY, USA, September 1998, IEEE Press
- JADE-Tilab (2009). Java Agent Development Framework, 2009, <http://jade.tilab.com>
- Jiang, Y.C.; Xia, Z.Y. & Zhang, S.Y. (2005). A novel defense model for dynamic topology network based on mobile agent. *Microprocessor and Microsystems*, 29, 6 (August 2005) 289-297
- Kim, J. S. (2002). Distributed Genetic Algorithm with Multiple Populations Using Multi-agent, *Proceedings of the 4th International Symposium on High Performance Computing*, pp. 329-334, London, Springer-Verlag, UK
- Krasnogor, N. & Smith, J. (2005). A tutorial for competent memetic algorithms: Model, taxonomy, and design issues. *IEEE Transaction on Evolutionary Computation*, 9, 5, (October 2005) 474-488, 1089-778X
- Lacomme, P.; Prins, C. & Ramdane-Chérif, W. (2001). A Genetic Algorithm for the Capacitated Arc Routing Problem and Its Extensions, *Proceedings of EvoWorkshops on Applications of Evolutionary Computing*, pp. 473-483, 3-540-41920-9, Springer-Verlag, London, UK
- Lacomme, P.; Prins, C. & Ramdane-Chérif, W. (2004). Competitive Memetic Algorithms for Arc Routing Problems. *Annals of Operations Research*, 131, 159-185
- Laguna, M. & Martí, R. (2003). *Scatter Search: Methodology and Implementations in C*, Kluwer Academic Publishers, 1-4020-7376-3, Boston
- Lange, D.B. & Oshima, M. (1998). Programming and deploying Java mobile agents with aglets, In: *CA: Addison-Wesley*, Menlo Park.
- Lange, D.B. & Oshima, M. (1999). Seven good reasons for mobile agents. *Communications of the ACM*, 42, 3, (March 1999) 88-89, 0001-0782
- Lee, L.C.; Ndumu, D.T. & Nwana, H.S. (1998). ZEUS: An Advanced Tool-Kit for Engineering Distributed Multi-Agent Systems, *Proceedings of Practical Application of Intelligent Agents and Multi-Agent Systems*, pp. 377-392, London, UK
- Leeand, K.L.; Yun, J. S. & J. o., G. K. (2003) MoCAAS: Auction agent system using a collaborative mobile agent in electronic commerce. *Expert Systems with Applications*, 24, 2, 183-187
- Lind, J. (2001). *Iterative software engineering for multiagent systems The MASSIVE Method*, Springer-Verlag New York, Inc., 3540421661, Secaucus, NJ, USA
- Macêdo, R.J.A. & Assis Silva, F.M. (2005). The mobile group approach for the coordination of mobile agents. *Journal of Parallel and Distributed Computing*, 65, 3, (March 2005) 275-288, 0743-7315
- Metropolis, N.; Rosenbluth, A. W.; Rosenbluth, M. N.; Teller, A. H. & Teller, E. (1953). Equation of State Calculations by Fast Computing Machines. *J. Chem. Phys.*, 21, 6, 1087-1092
- Mladenović, N. & Hansen, P. (1997). Variable neighborhood search. *Computers & Operations Research*, 24, 11, November 1997, 1097-1100
- O'Brien, P. & Nicol, R. (1998). FIPATowards a standard for software agents. *BT Technology Journal*, 16, 3 (1998) 51-59
- Papavassiliou, S.; Puliafito, A.; Tomarchio, O. & Ye, J. (2002). Mobile agent-based approach for efficient network management and resource allocation: Framework and

- applications. *IEEE Journal of Selected Areas in Communication*, 20, 4 (May 2002) 858-872.
- Steen, V. S. & Tanenbaum, A. (2002). *Distributed Systems: Principles and Paradigms*, In: *Englewood Cliffs, NJ: Prentice Hall*
- Thati, P.; Chang, P. H. & Agha, G. (2001). Crawlets: Agents for high performance web search engine, *Proceedings of international conference on mobile agents*, pp. 119-134, 978-3-540-42952-4, January 2001, Springer Berlin / Heidelberg, Germany
- Ulusoy, G. (1985). The feet size and mix problem for capacitated arc routing. *European Journal of Operational Research*, 22, 3, 329-337.
- Veldhuizen, D. A.; Zydallys, J. B. & Lamont, G. B. (2003). Considerations in engineering parallel multiobjective evolutionary algorithms. *IEEE Transactions on Evolutionary Computation*, 7, 2, (April 2003) 144-173, 1089-778X
- White, J. E. (1997). Mobile Agents, In: *Software Agents*, Bradshaw, J. (Ed.), MA: MIT Press, Cambridge
- WIDE (2009). Wide user guide, <http://jade.tilab.com/wade>

11. Appendix - List of Symbols and Acronyms

AM	Augment-Merge
AOP	Agent Oriented Programming
CARP	Capacitated Arc Routing problem
CORBA	Common Object Request Broker Architecture
CPU	Central Processing Unit
CVRP	Capacitated Vehicle Routing Problem
DAIE	Distributed Algorithm with Individuals exchange
DCE	Distributed Computing Environment
DCOM	Distributed Component Object Model
DM-MIMD	Distributed Memory Multiple Instruction stream, Multiple Data stream
EA	Evolutionary Algorithm
EC-code	Evolutionary Computation code
FIPA	Foundation for Intelligent Physical Agents
GA	Genetic Algorithm
JADE	Java Agent DEvelopment Framework
LOX	Linear Order Crossover
LS	Local Search
LSA	Local Status Agent
MA	Mobile Agent
MAS	Multi Agent System
MOM	Message-Oriented Middleware
MSA	Main Status Agent
OOP	Object Oriented Programming
OX	Order Crossover
PS	Path Scanning
RPC	Remote Procedure Call
RMI	Remote Method Invocation
SA	Status Agent

SPA	Static Parallel Algorithm
SYNCA	SYNChronization Agent
UH	Ulusoy's Heuristic
UIA	User Interface Agent
UWCOP	Underground Waste Collection Optimization Problem
X1	One point Crossover

IntechOpen

IntechOpen

Contemporary Robotics - Challenges and Solutions

Edited by A D Rodi

ISBN 978-953-307-038-4

Hard cover, 392 pages

Publisher InTech

Published online 01, December, 2009

Published in print edition December, 2009

This book is a collection of 18 chapters written by internationally recognized experts and well-known professionals of the field. Chapters contribute to diverse facets of contemporary robotics and autonomous systems. The volume is organized in four thematic parts according to the main subjects, regarding the recent advances in the contemporary robotics. The first thematic topics of the book are devoted to the theoretical issues. This includes development of algorithms for automatic trajectory generation using redundancy resolution scheme, intelligent algorithms for robotic grasping, modelling approach for reactive mode handling of flexible manufacturing and design of an advanced controller for robot manipulators. The second part of the book deals with different aspects of robot calibration and sensing. This includes a geometric and threshold calibration of a multiple robotic line-vision system, robot-based inline 2D/3D quality monitoring using picture-taking and laser triangulation, and a study on prospective polymer composite materials for flexible tactile sensors. The third part addresses issues of mobile robots and multi-agent systems, including SLAM of mobile robots based on fusion of odometry and visual data, configuration of a localization system by a team of mobile robots, development of generic real-time motion controller for differential mobile robots, control of fuel cells of mobile robots, modelling of omni-directional wheeled-based robots, building of hunter- hybrid tracking environment, as well as design of a cooperative control in distributed population-based multi-agent approach. The fourth part presents recent approaches and results in humanoid and bioinspirative robotics. It deals with design of adaptive control of anthropomorphic biped gait, building of dynamic-based simulation for humanoid robot walking, building controller for perceptual motor control dynamics of humans and biomimetic approach to control mechatronic structure using smart materials.

How to reference

In order to correctly reference this scholarly work, feel free to copy and paste the following:

Kamel Belkheili, Pierre Chauvet and Arnaud Schaal (2009). Cooperation Control in Distributed Population-based Algorithms Using a Multi-Agent Approach Application to a Real-Life Vehicle Routing Problem, Contemporary Robotics - Challenges and Solutions, A D Rodi (Ed.), ISBN: 978-953-307-038-4, InTech, Available from: <http://www.intechopen.com/books/contemporary-robotics-challenges-and-solutions/cooperation-control-in-distributed-population-based-algorithms-using-a-multi-agent-approach-application>

INTECH
open science | open minds

InTech Europe

InTech China

www.intechopen.com

University Campus STeP Ri
Slavka Krautzeka 83/A
51000 Rijeka, Croatia
Phone: +385 (51) 770 447
Fax: +385 (51) 686 166
www.intechopen.com

Unit 405, Office Block, Hotel Equatorial Shanghai
No.65, Yan An Road (West), Shanghai, 200040, China
中国上海市延安西路65号上海国际贵都大饭店办公楼405单元
Phone: +86-21-62489820
Fax: +86-21-62489821

IntechOpen

IntechOpen

© 2009 The Author(s). Licensee IntechOpen. This chapter is distributed under the terms of the [Creative Commons Attribution-NonCommercial-ShareAlike-3.0 License](https://creativecommons.org/licenses/by-nc-sa/3.0/), which permits use, distribution and reproduction for non-commercial purposes, provided the original is properly cited and derivative works building on this content are distributed under the same license.

IntechOpen

IntechOpen